Testing & Deploying Microservices

Sam Newman

ThoughtWorks®

TECHNOLOGY RADAR

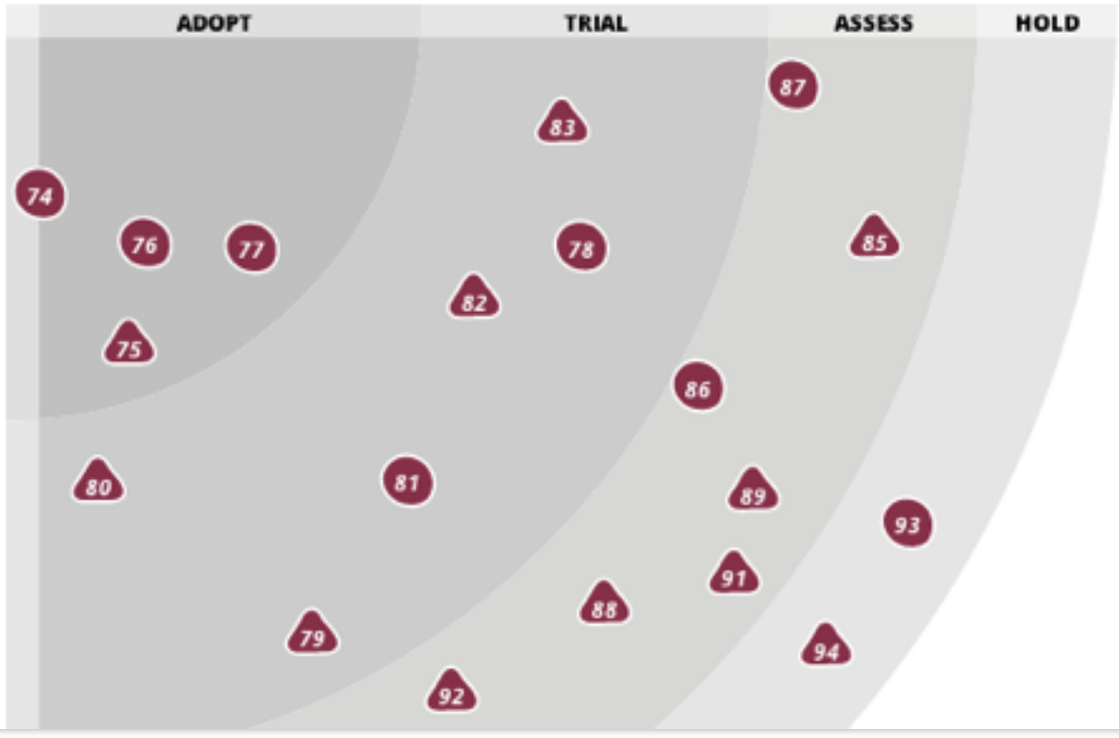
Radar home Techniques Tools Platforms Languages & frameworks Radar A-Z

ADOPT

74. Clojure
75. Dropwizard
76. Scala, the good parts
77. Sinatra

TRIAL

78. CoffeeScript
79. Go language NEW
80. Hive NEW
81. Play Framework 2
82. Reactive Extensions across languages NEW
83. Web API NEW



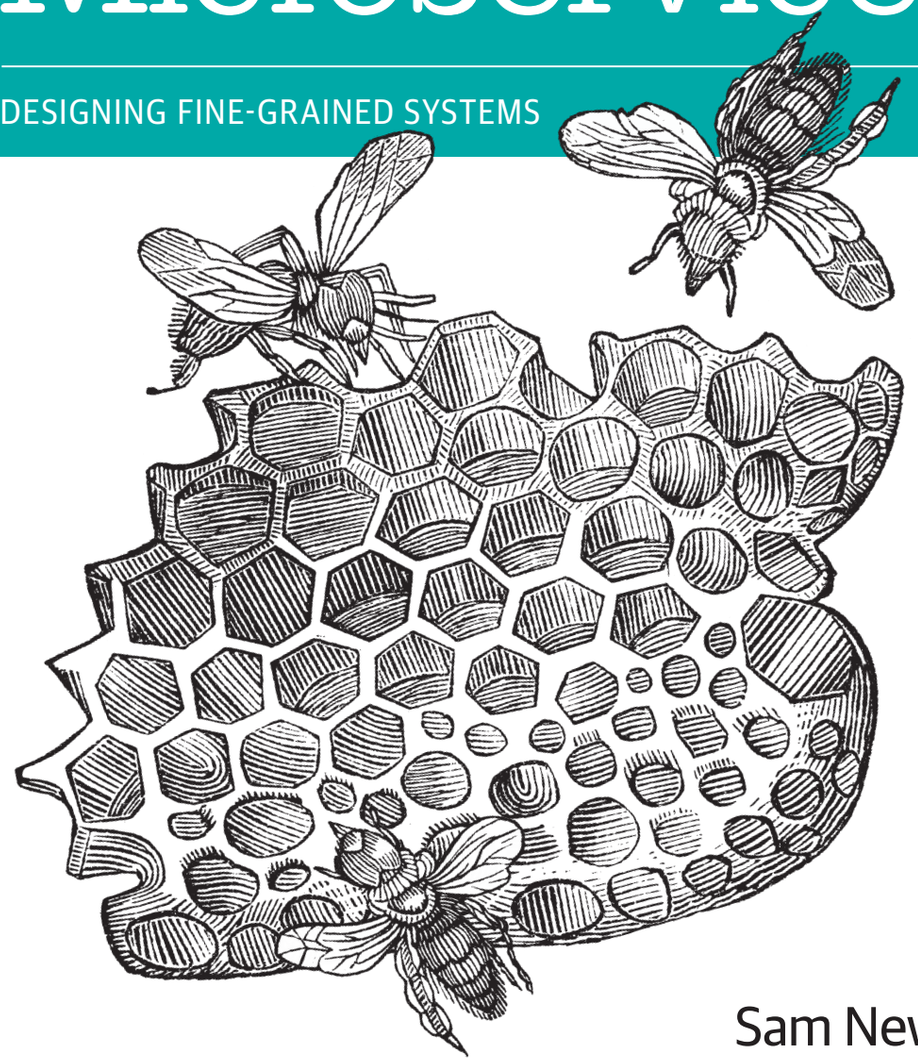
http://www.thoughtworks.com/radar/

#geecon @samnewman

O'REILLY®

Building Microservices

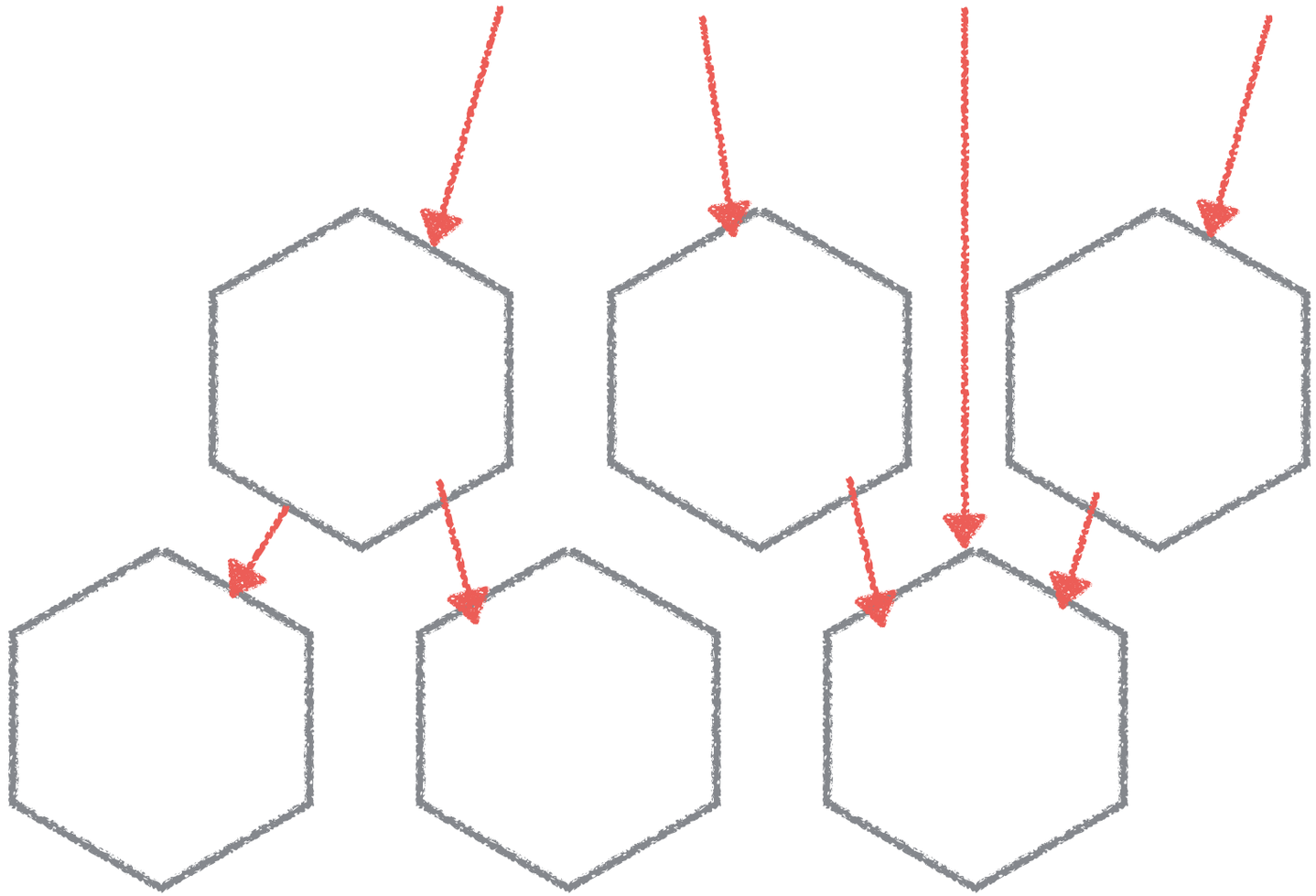
DESIGNING FINE-GRAINED SYSTEMS

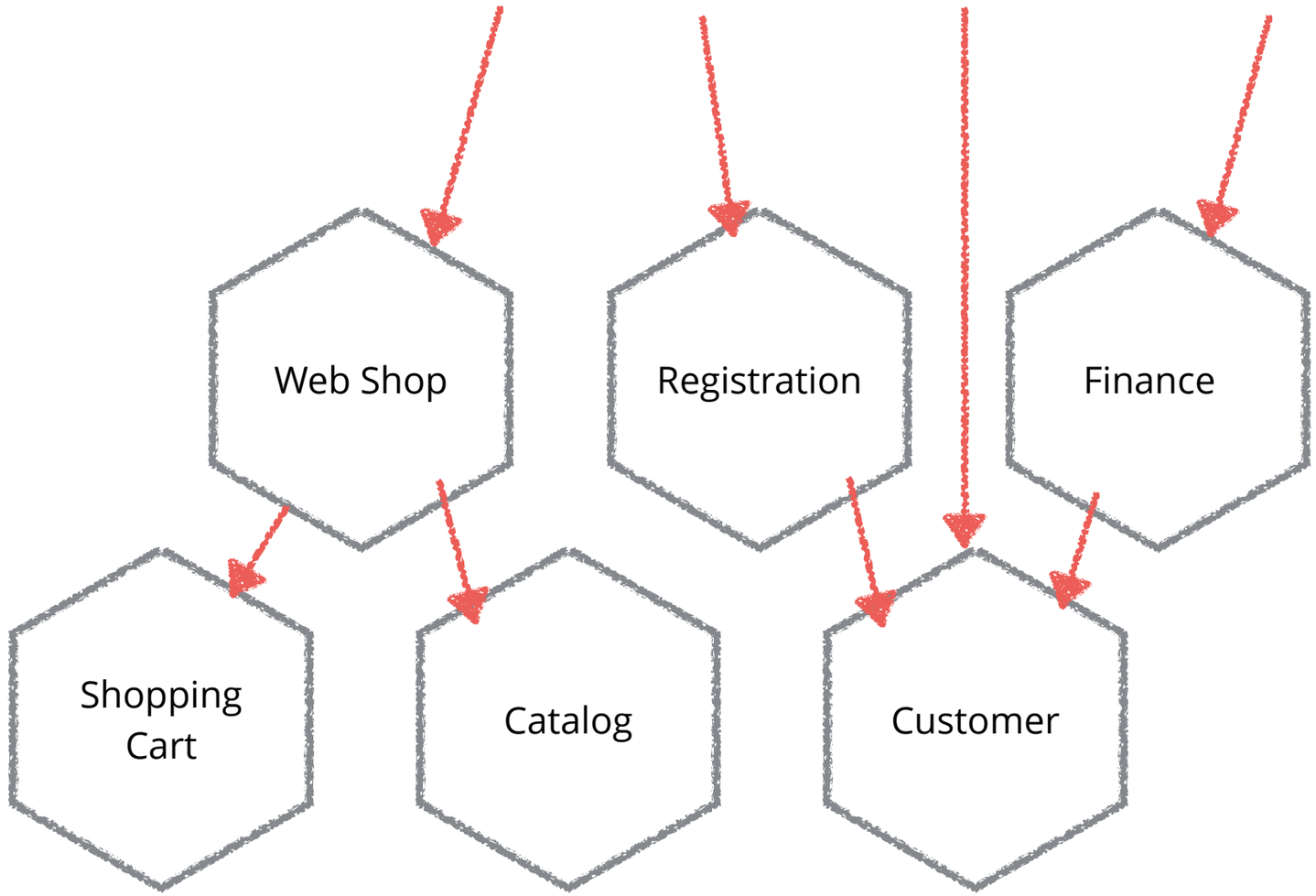


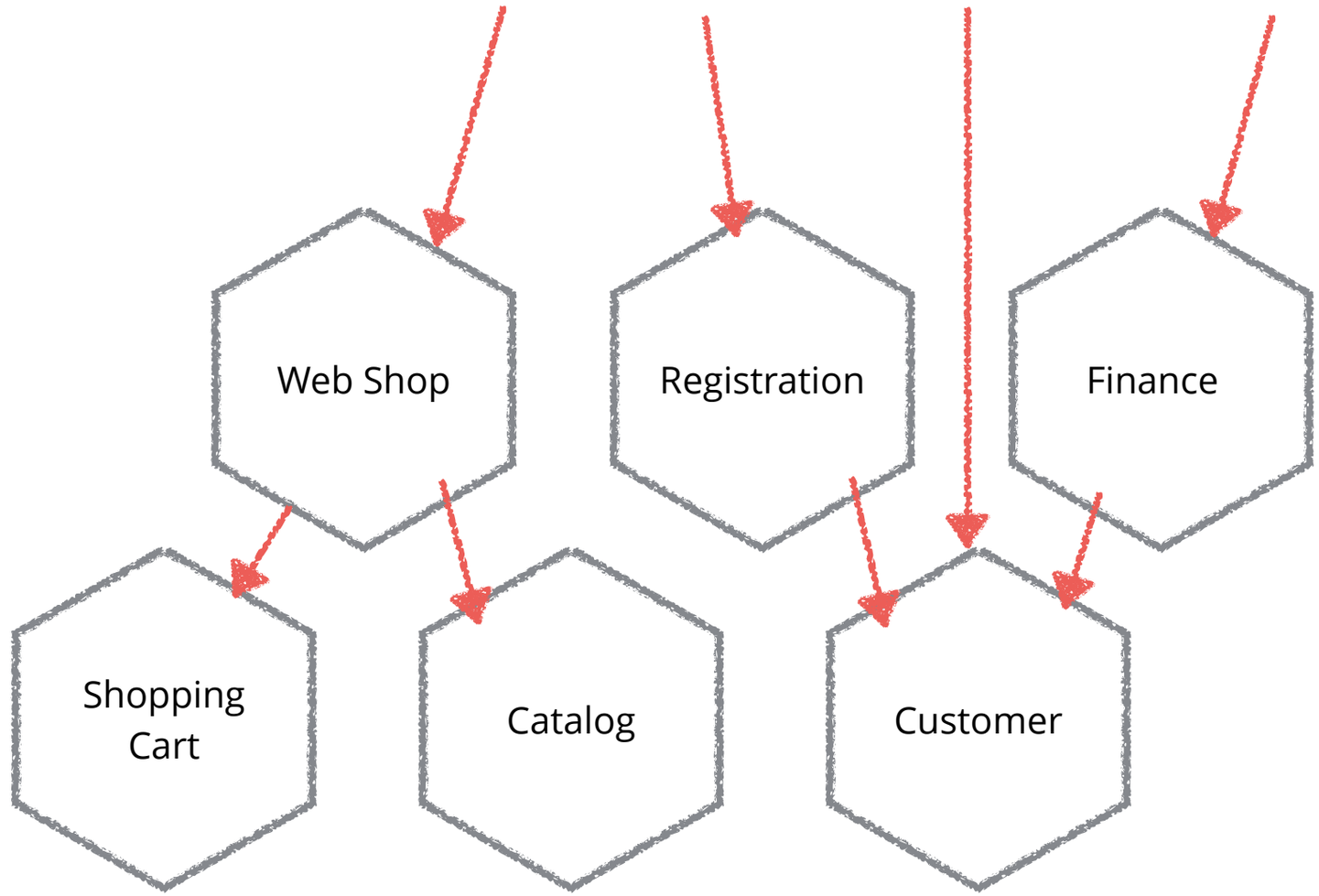
Sam Newman

#geecon

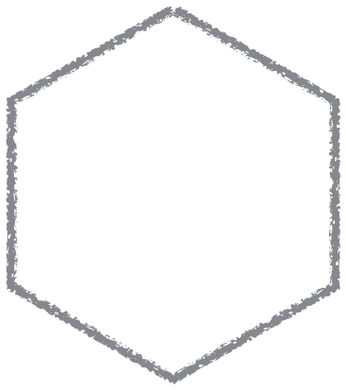
@samnewman

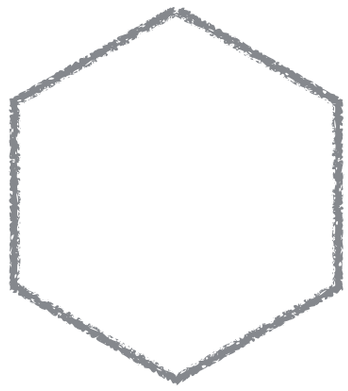


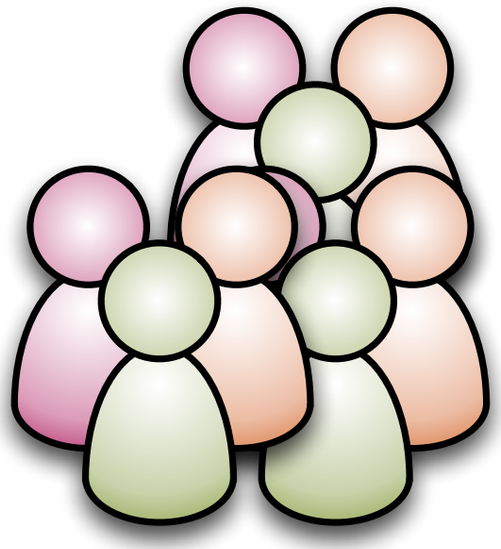

Web Shop
Registration
Finance
Shopping Cart
Catalog
Customer

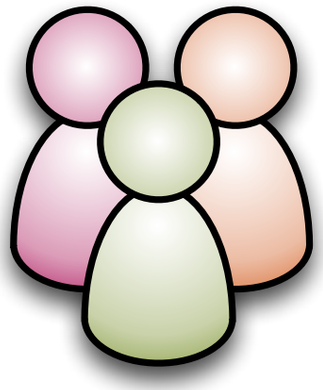


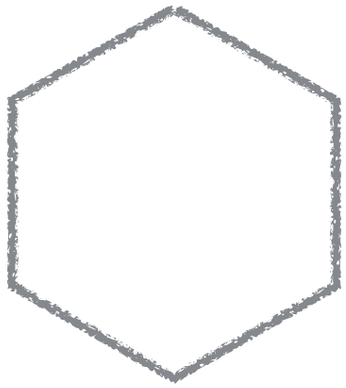
1000 lines of code or less

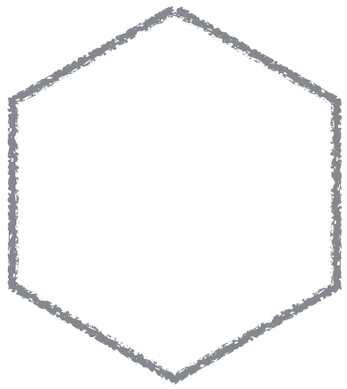


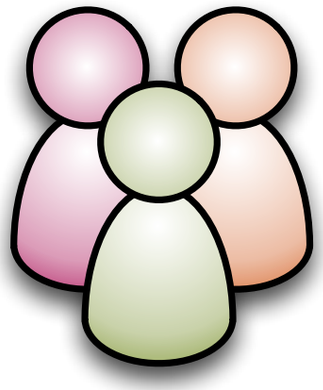






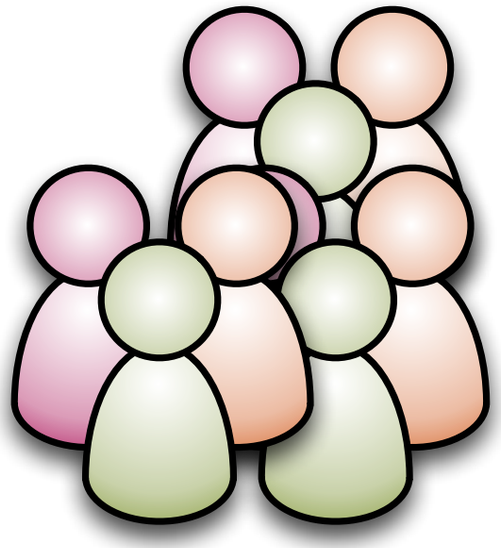


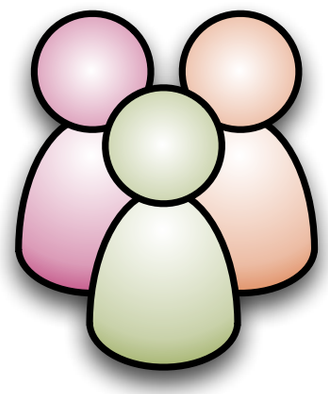


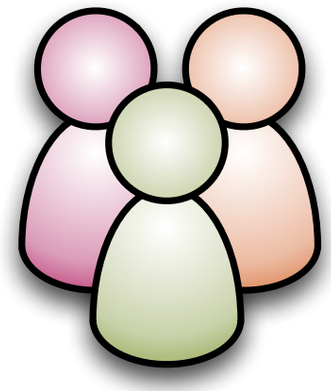


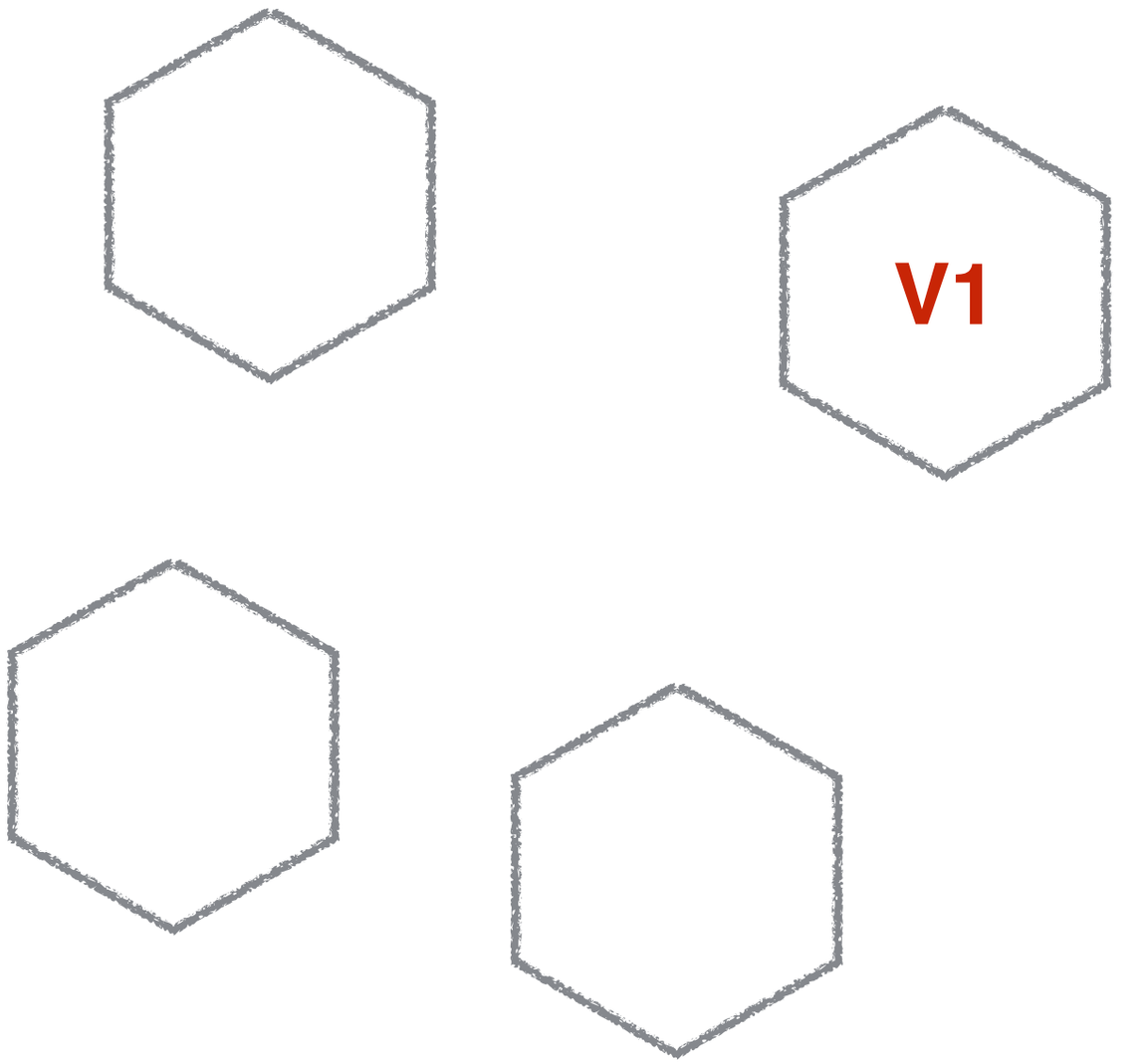
#geecon

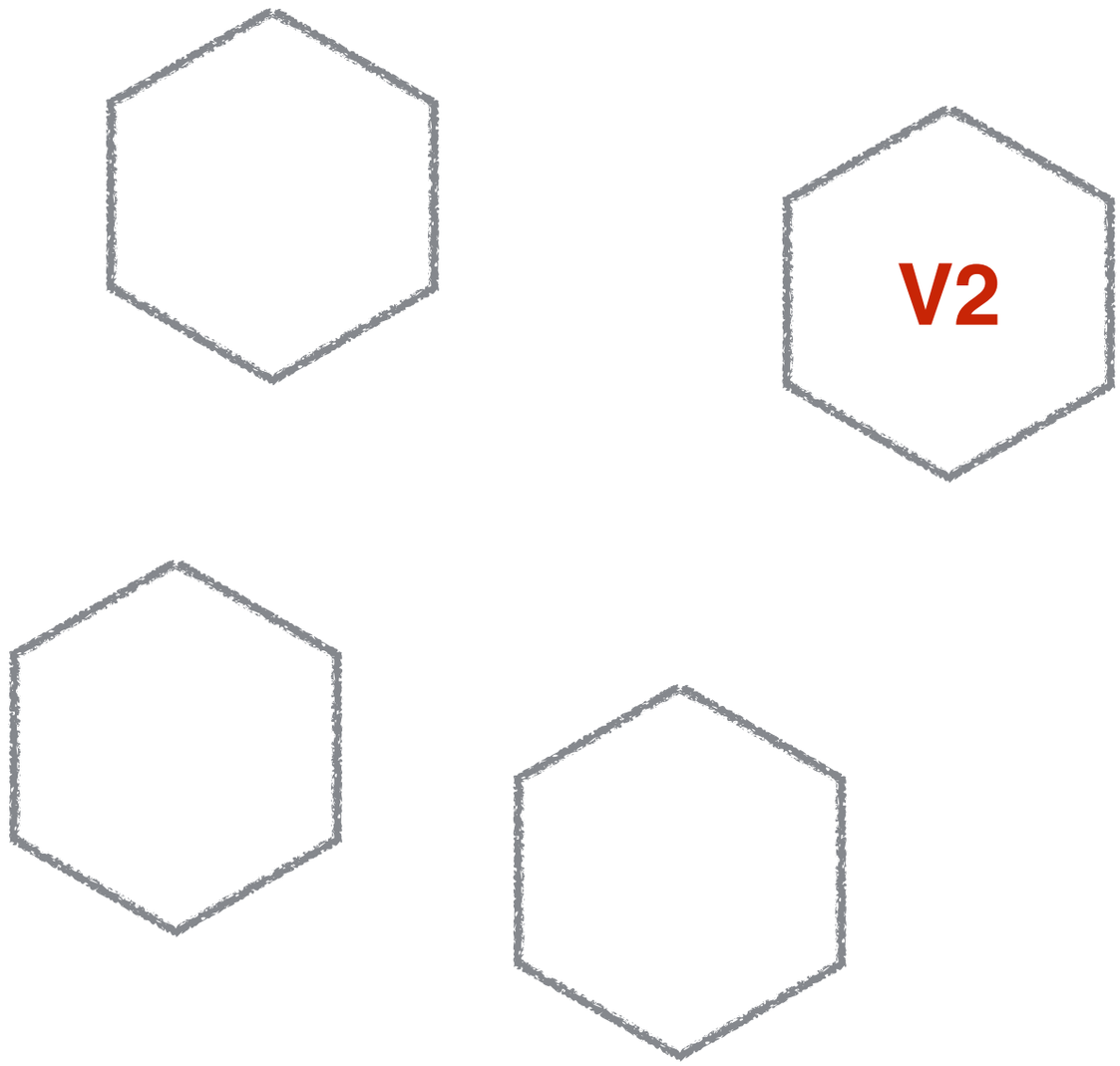
@samnewman

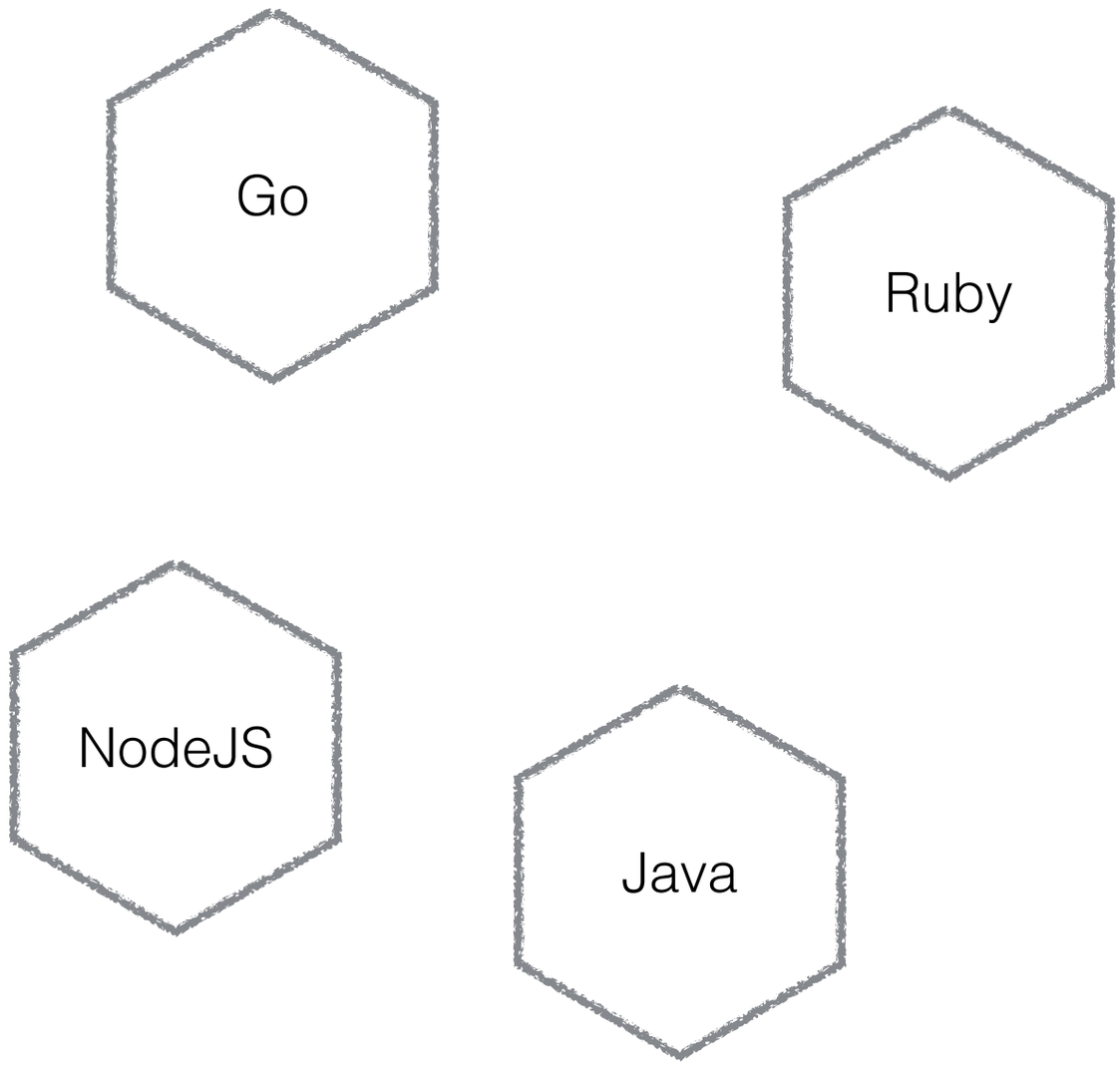


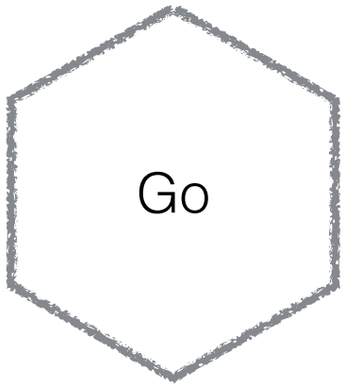


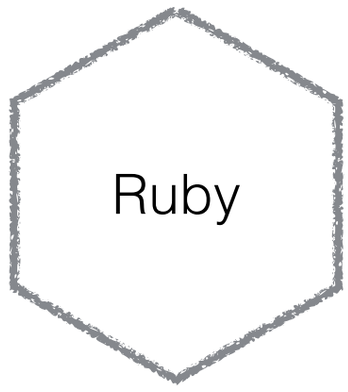


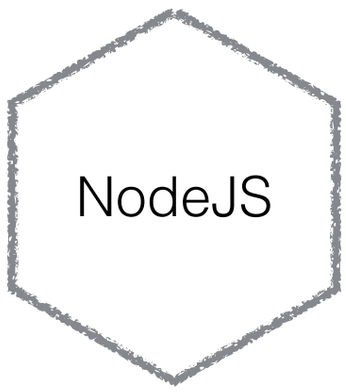

V1

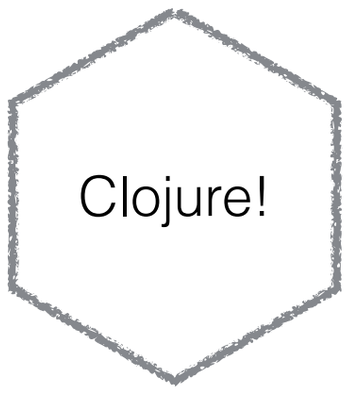

V2

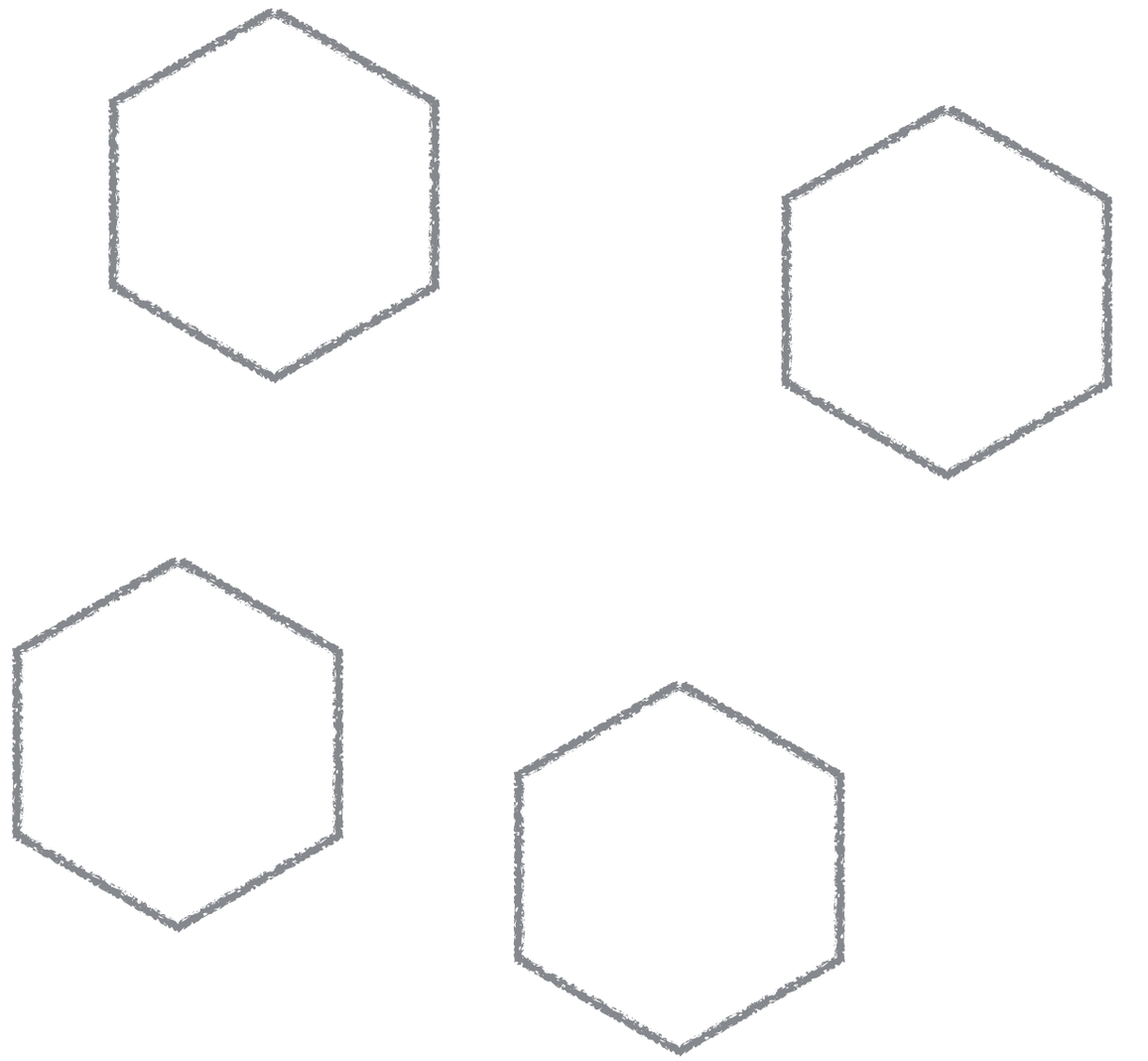

Go
Ruby
NodeJS
Java

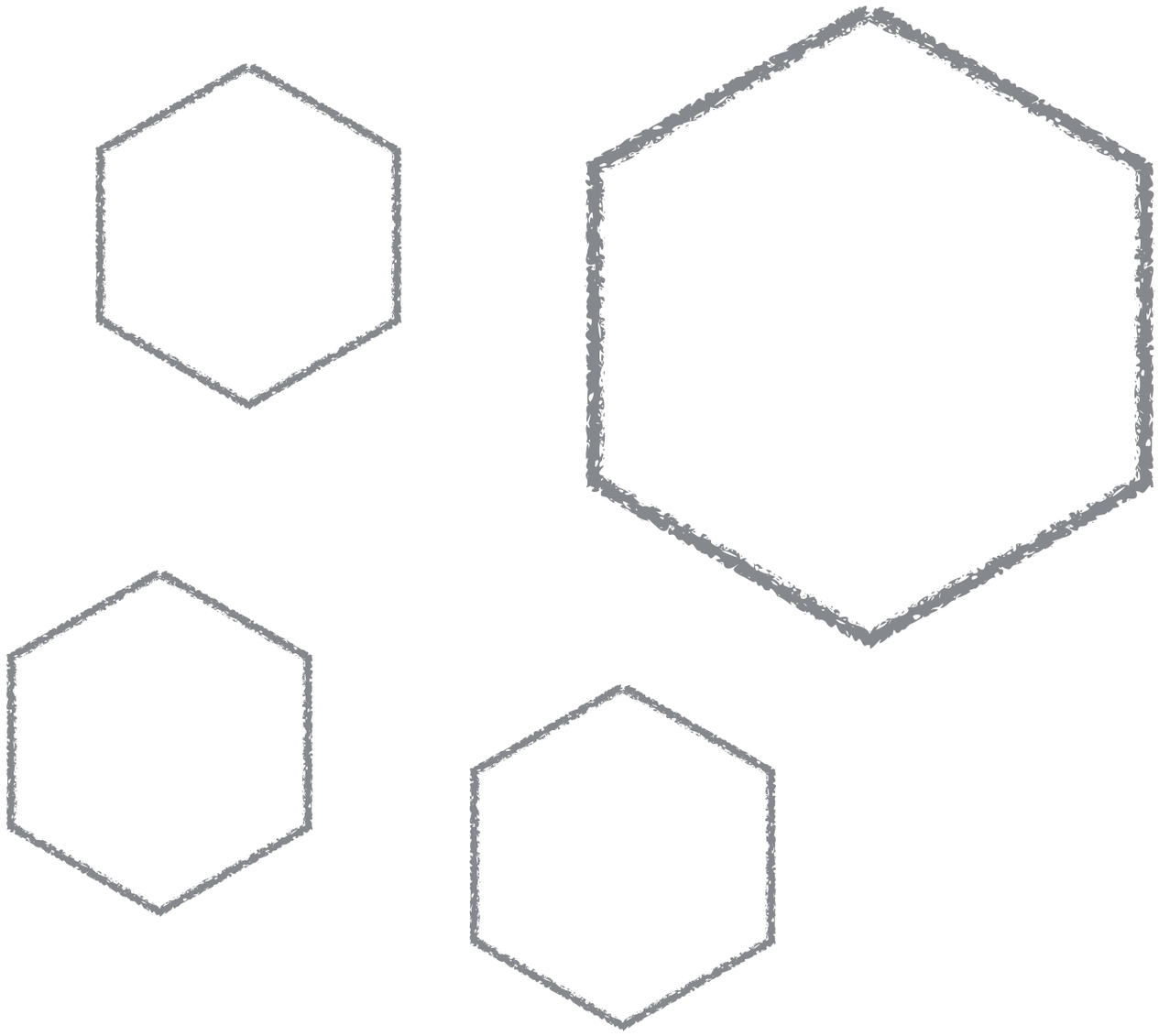

Go

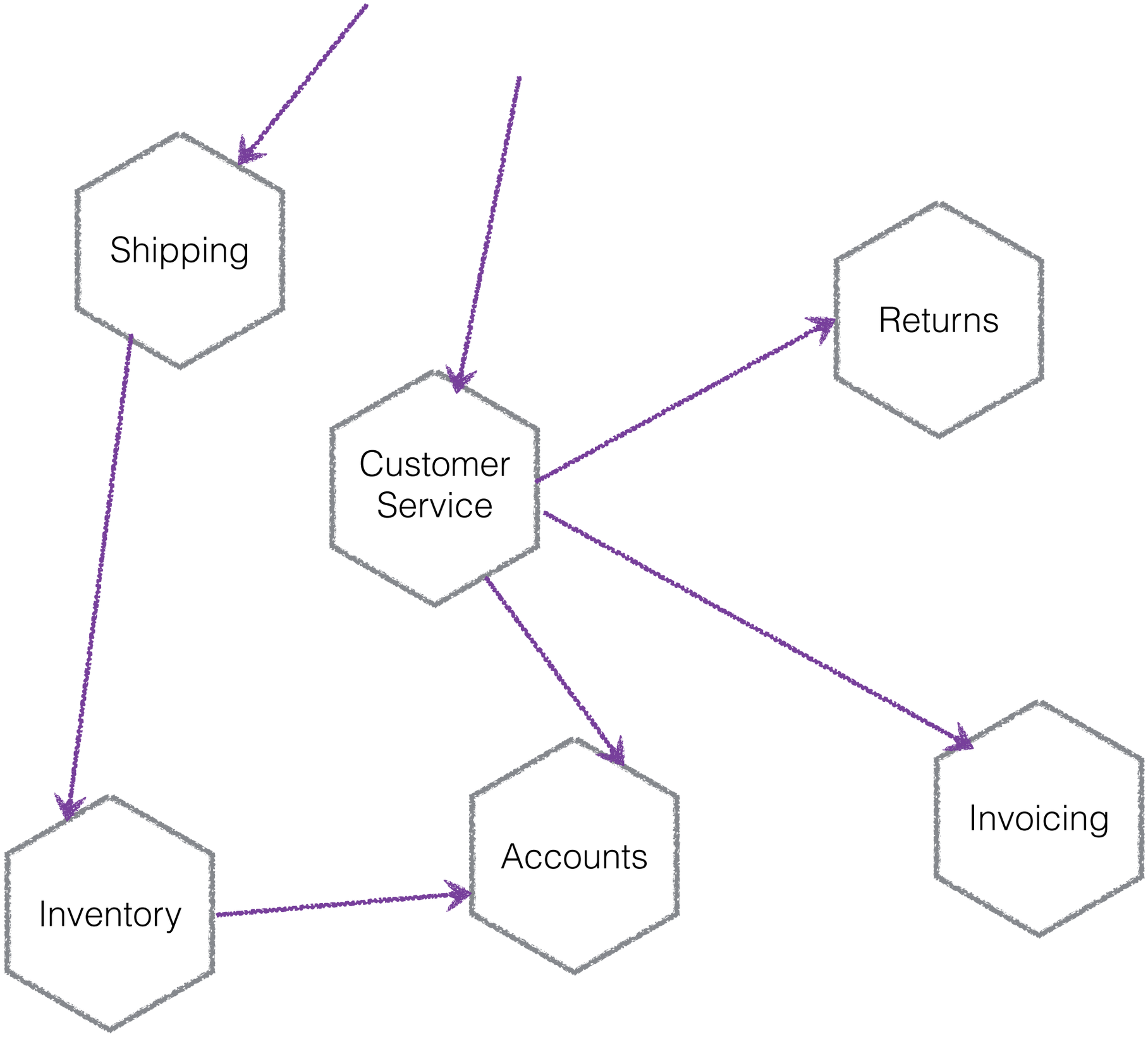

Ruby

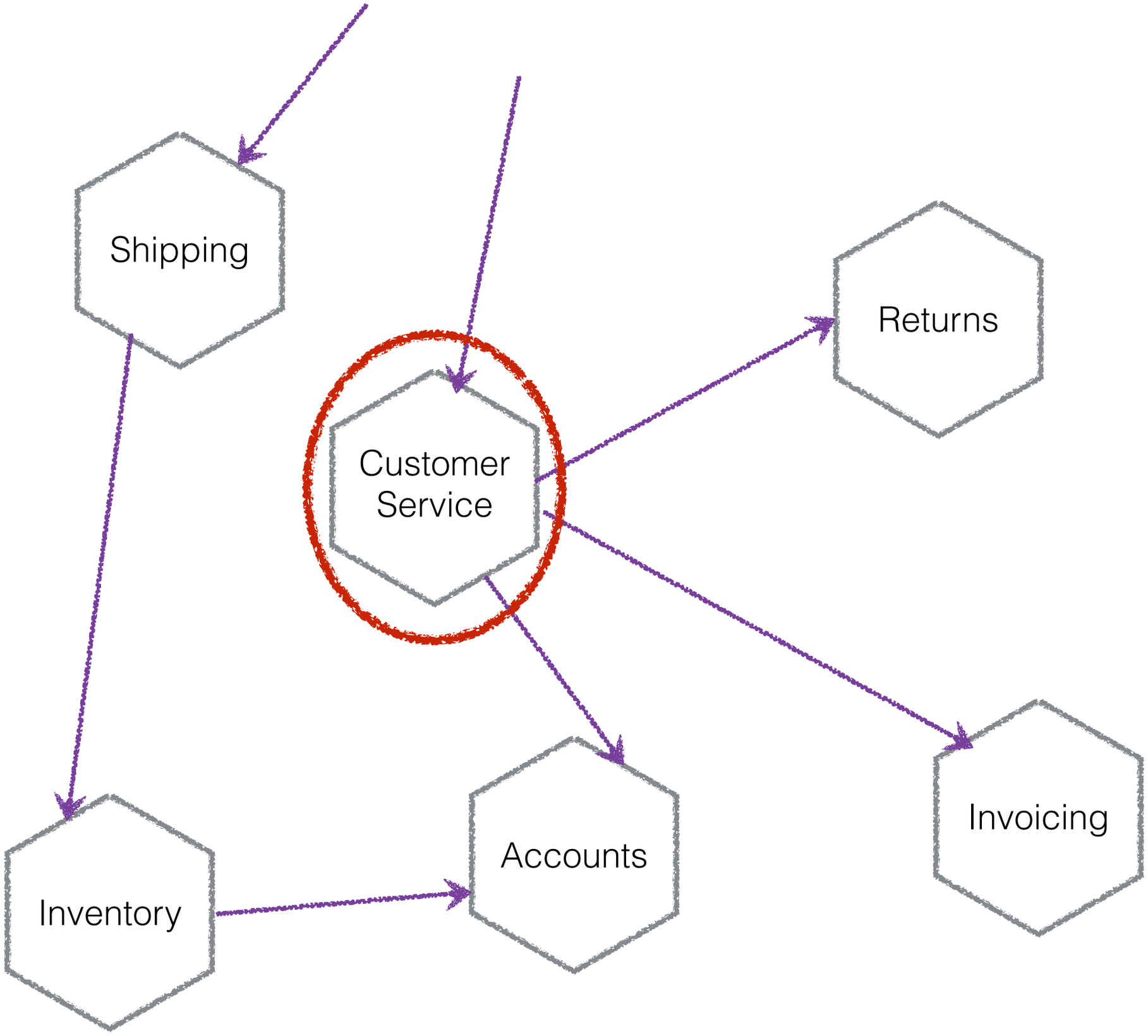

NodeJS

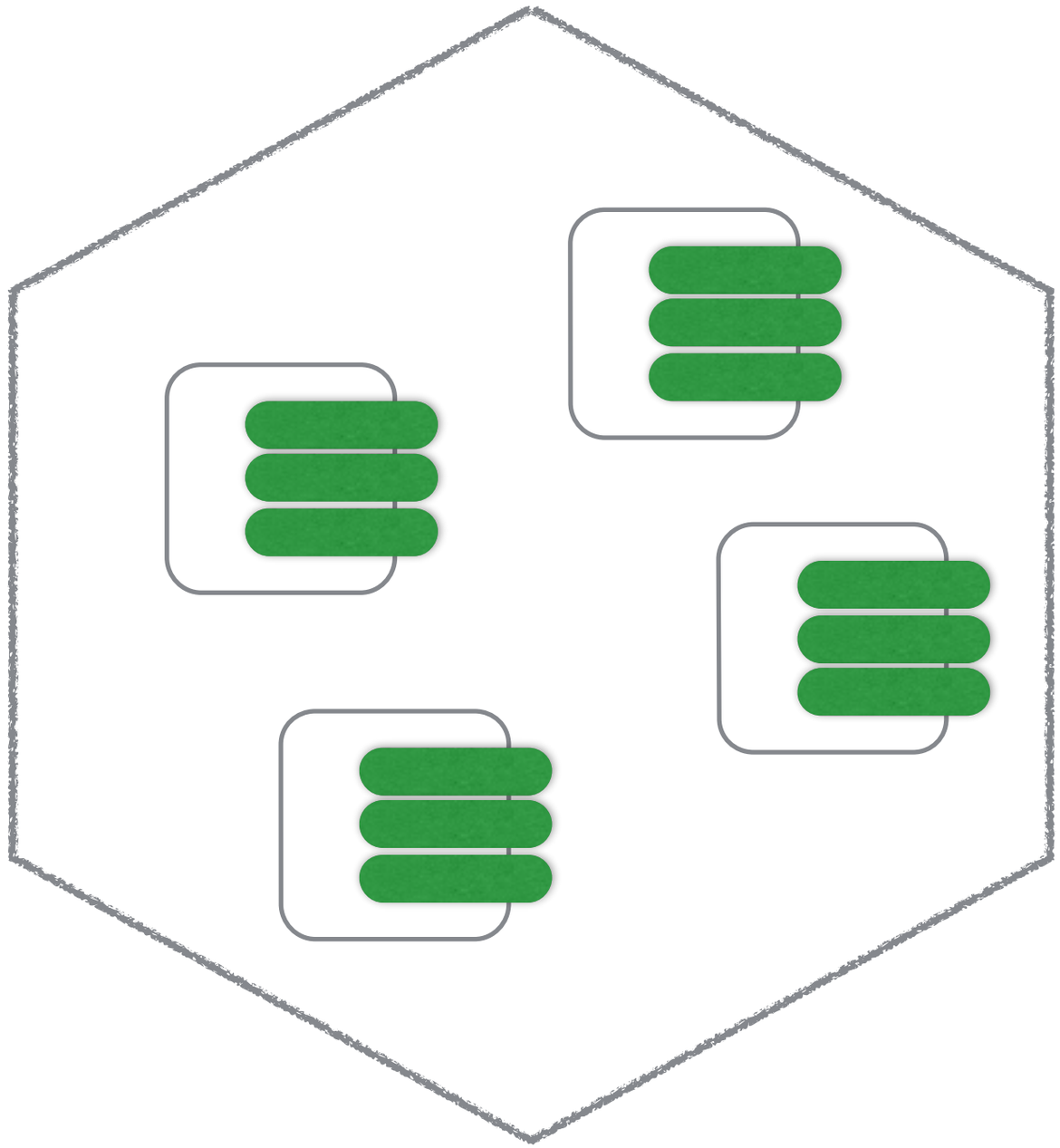

Clojure!

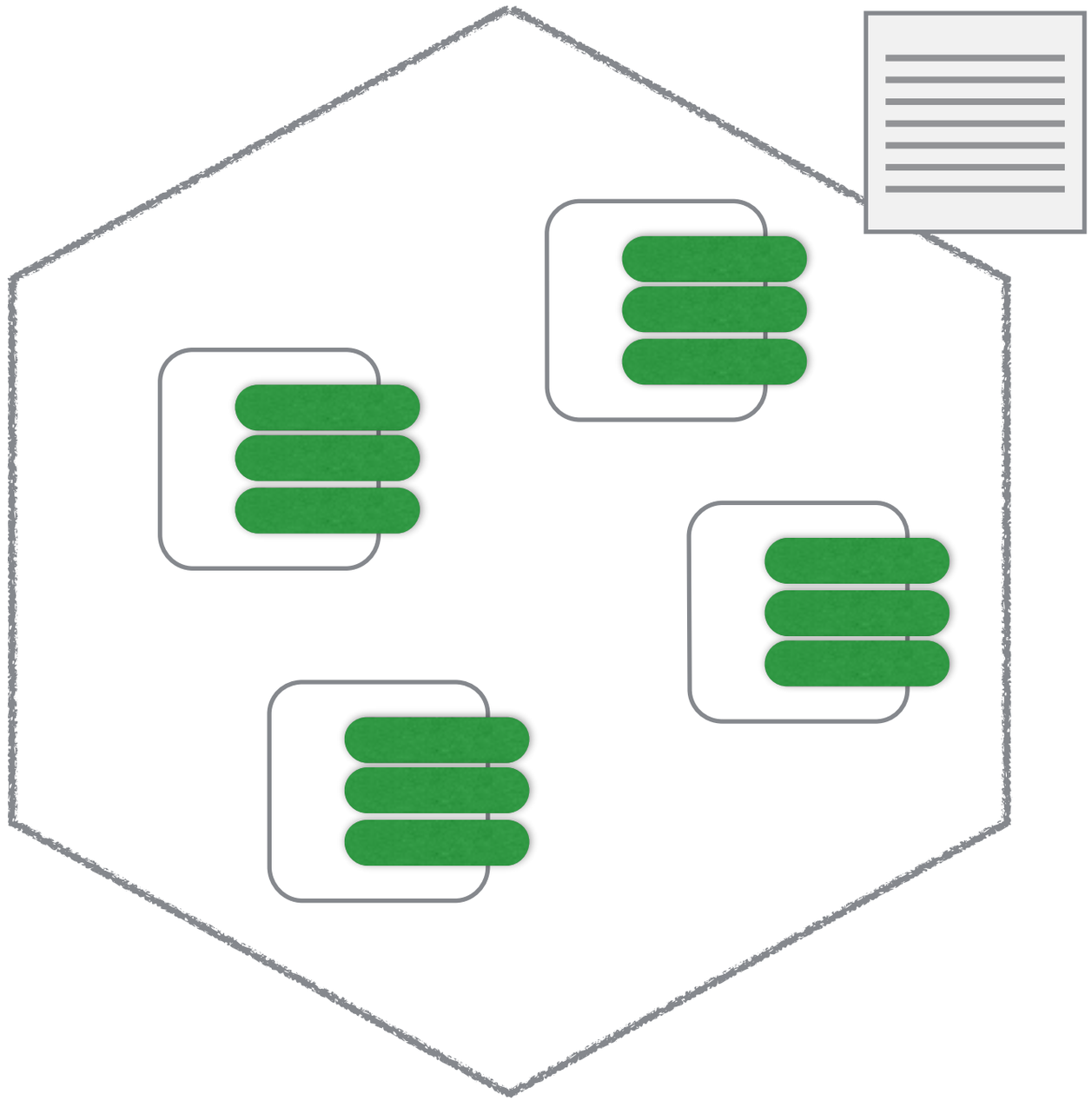


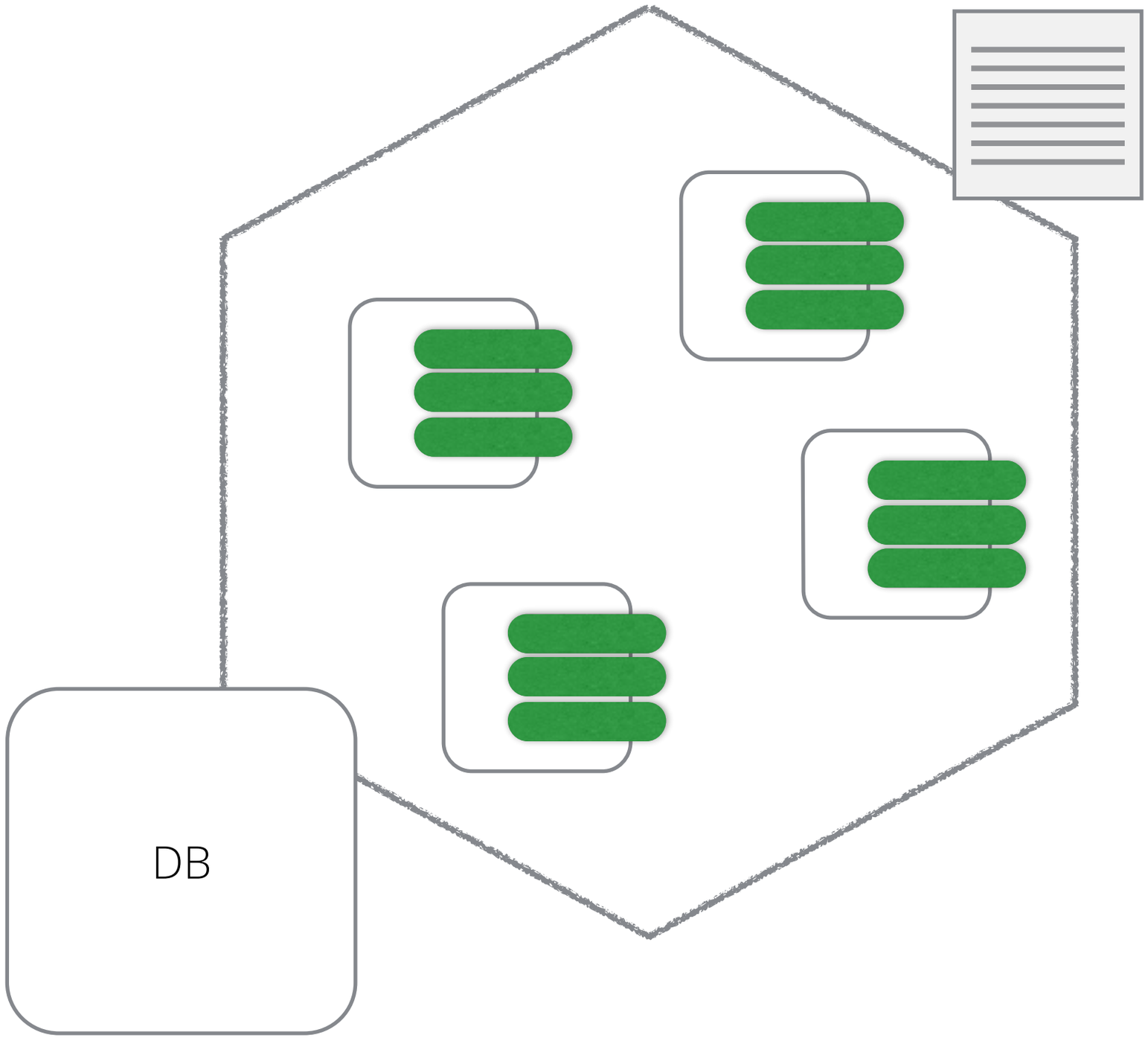



Shipping
Customer Service
Returns
Inventory
Accounts
Invoicing

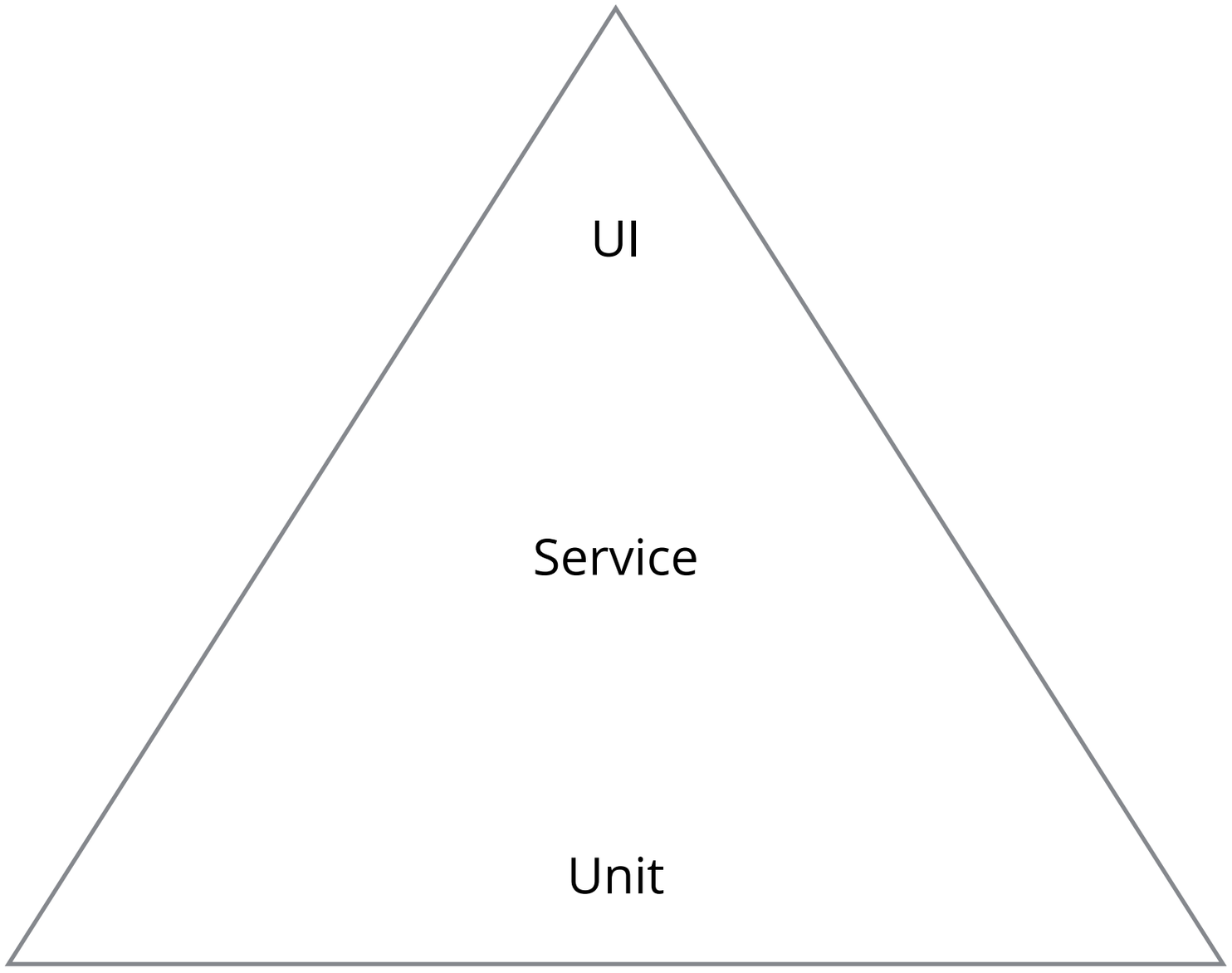

Shipping
Customer Service
Returns
Inventory
Accounts
Invoicing



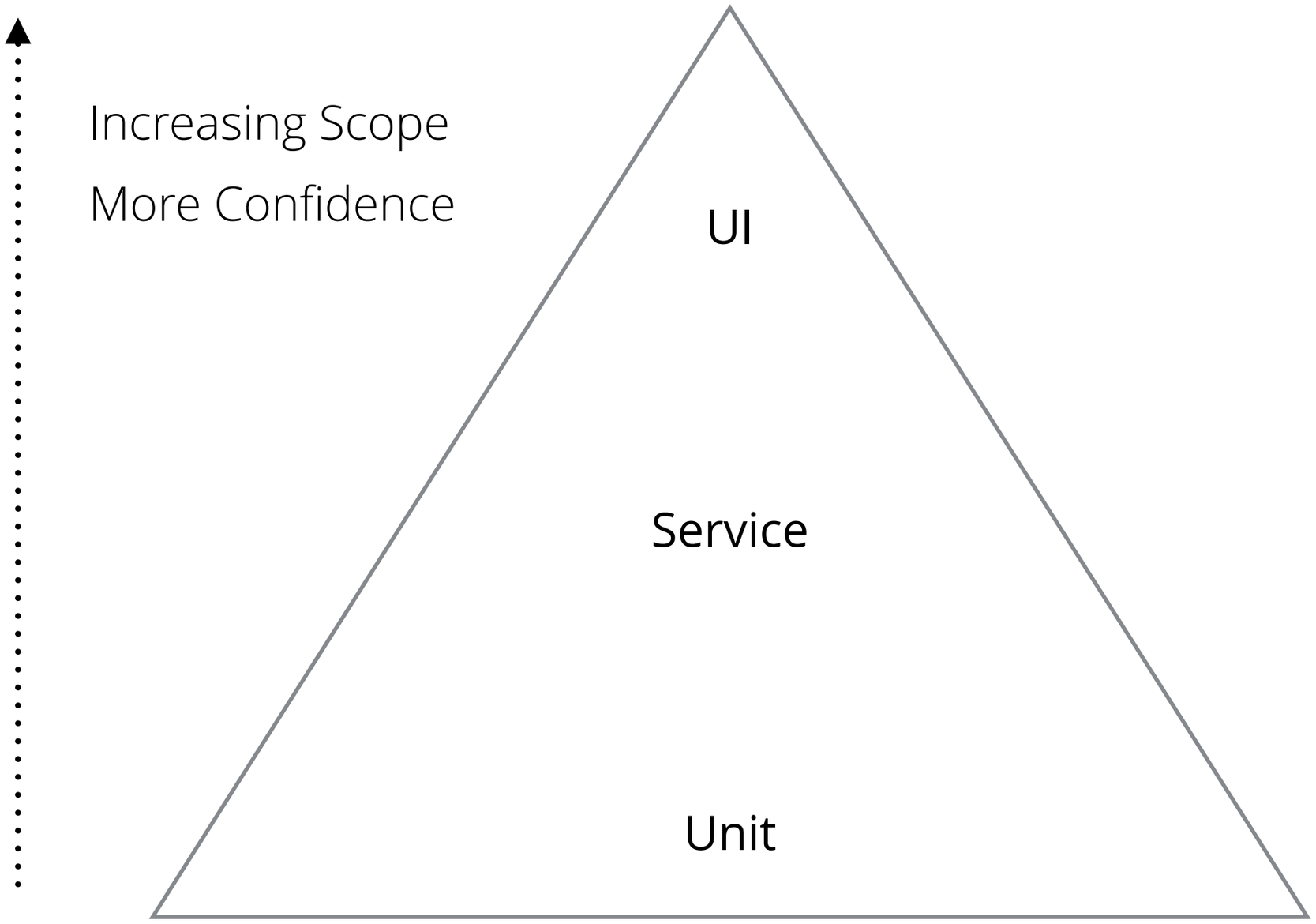



DB

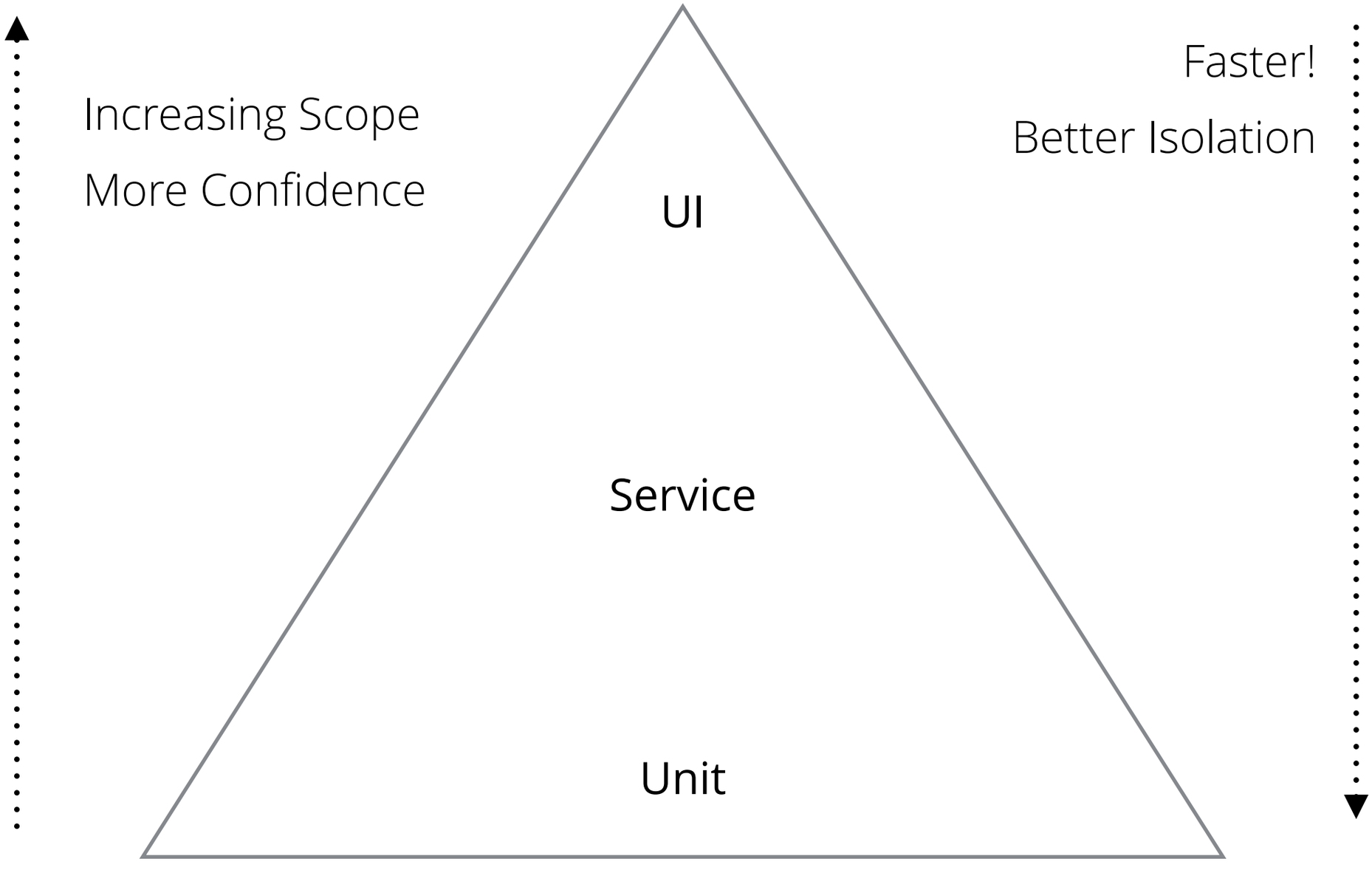
MIKE COHN'S TEST PYRAMID



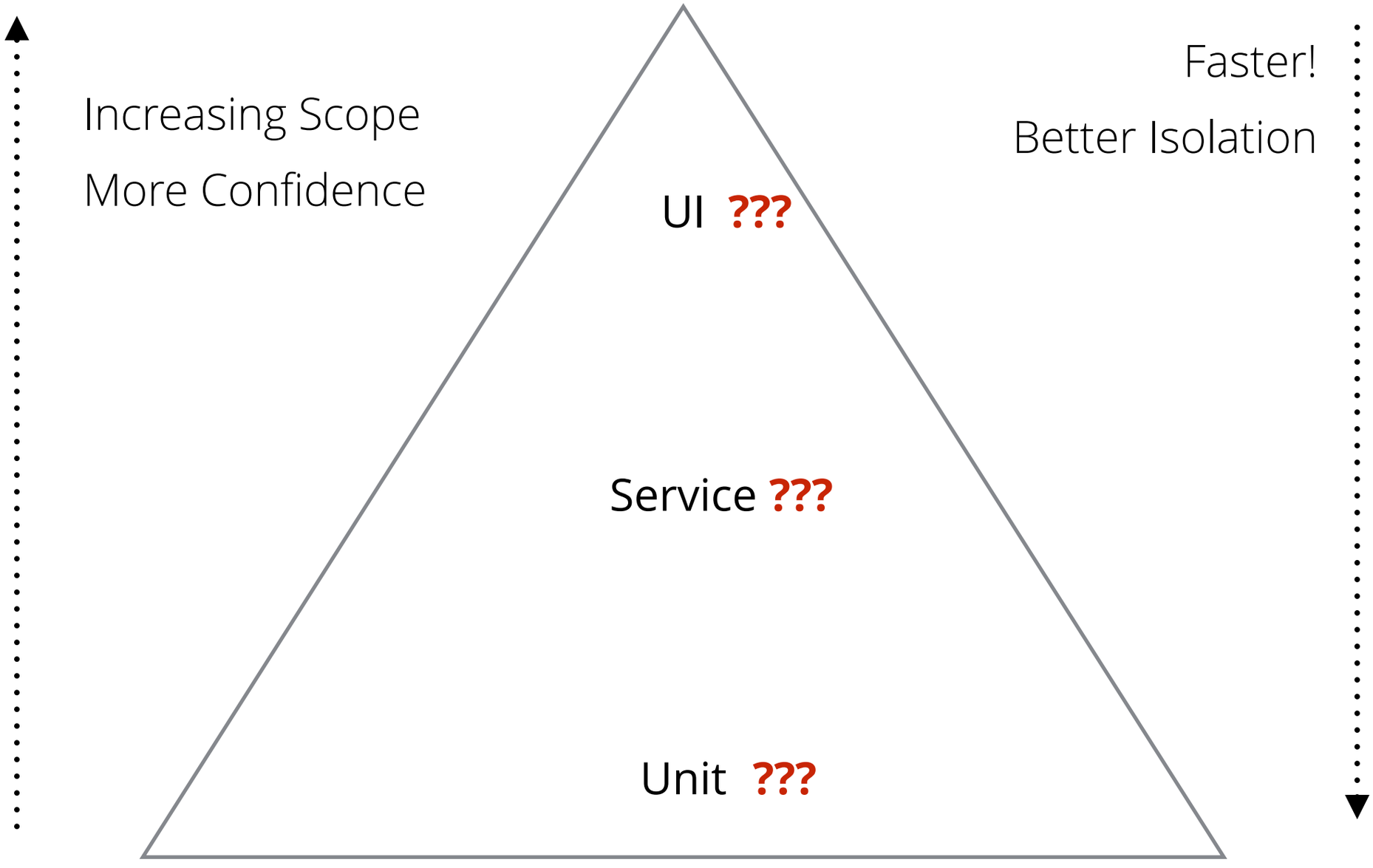
MIKE COHN'S TEST PYRAMID



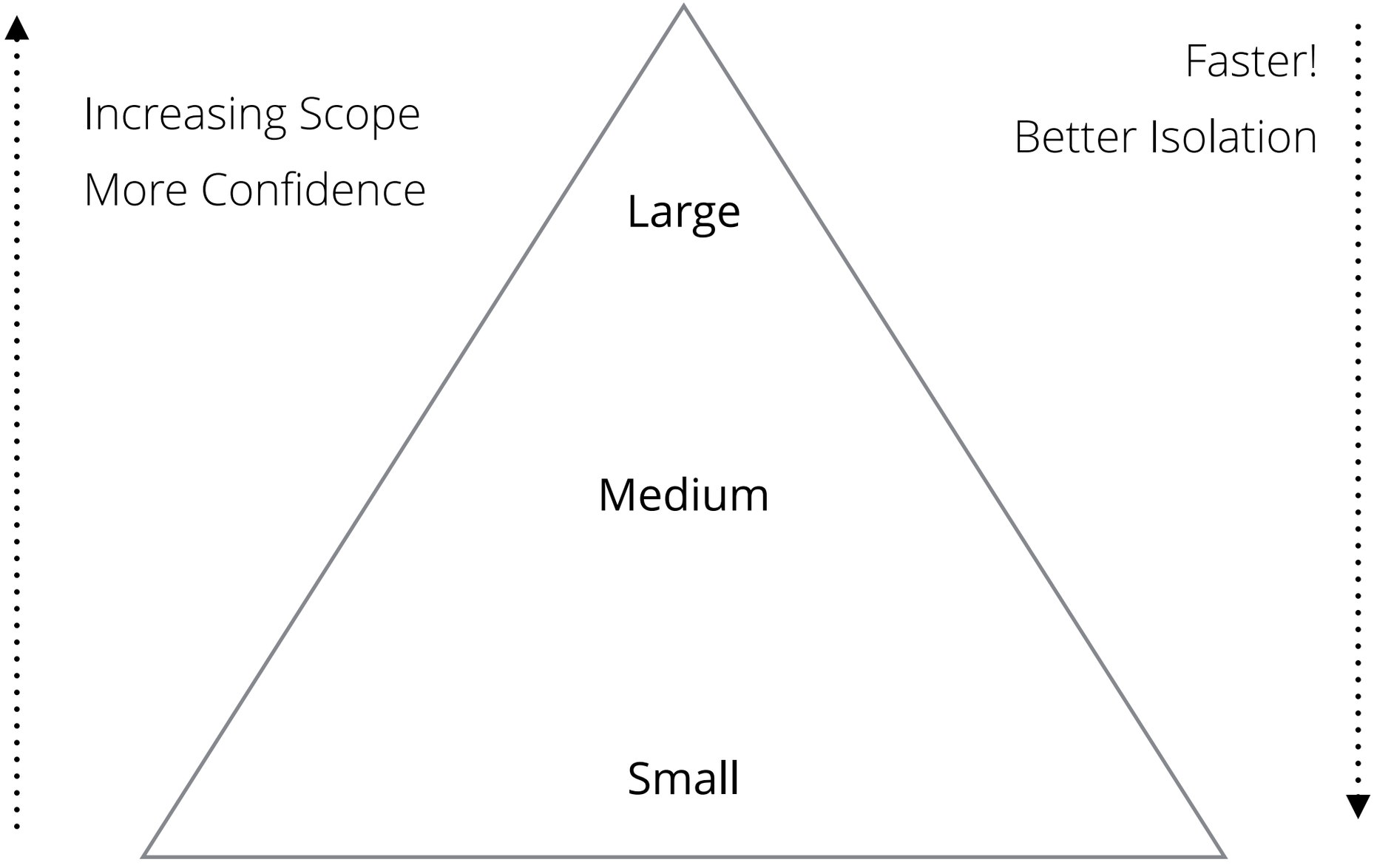
MIKE COHN'S TEST PYRAMID

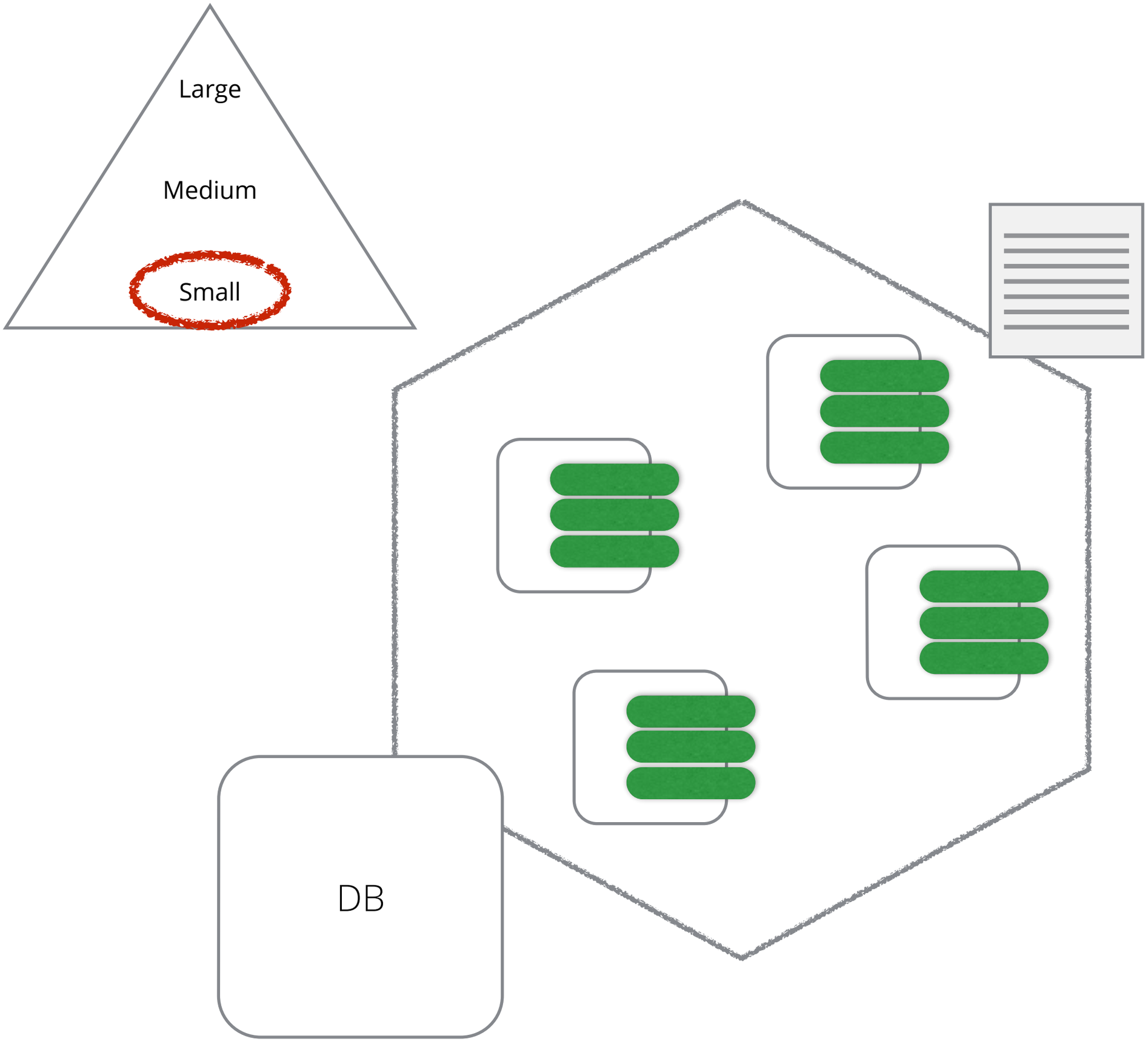


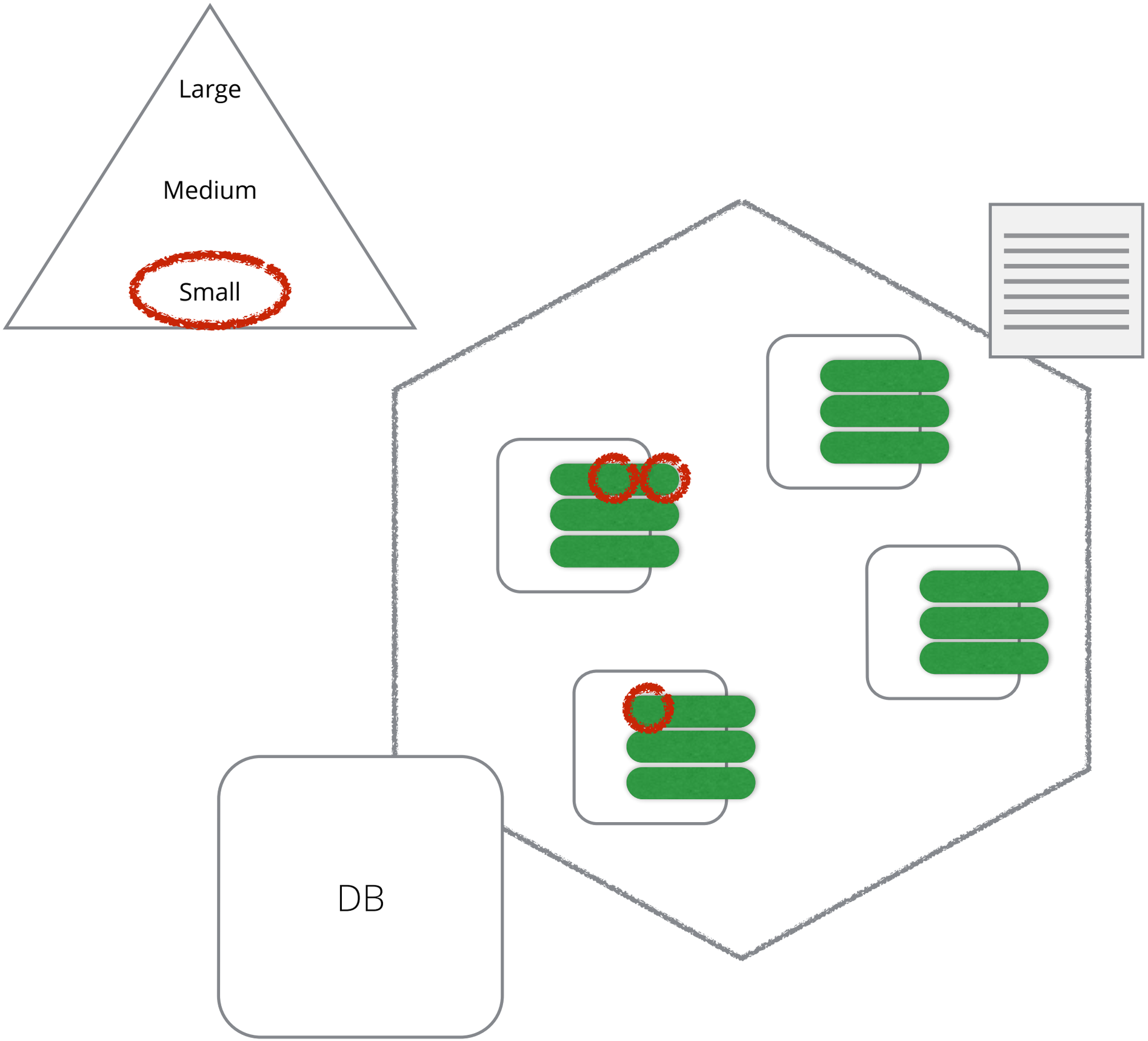
MIKE COHN'S TEST PYRAMID



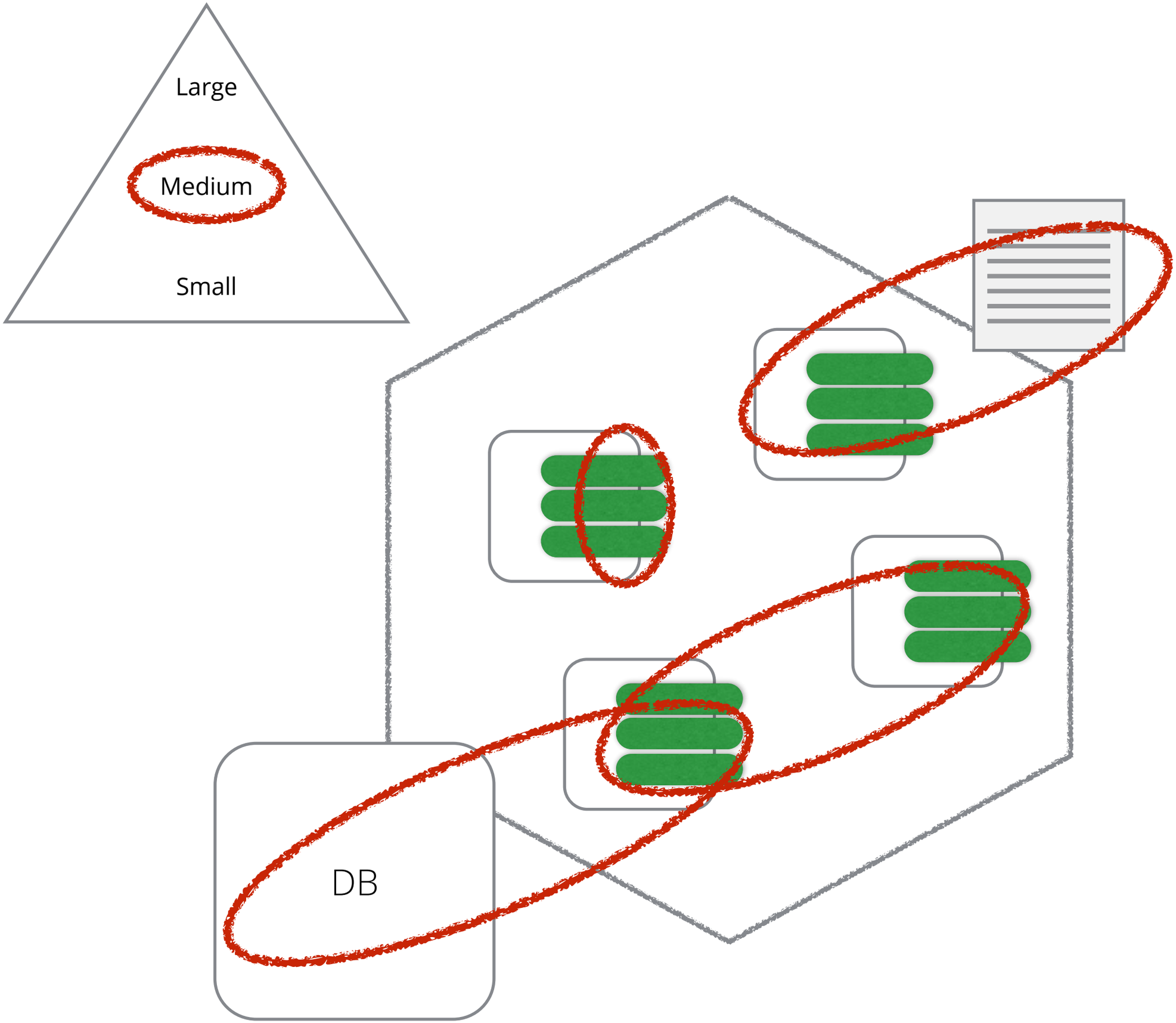
GOOGLE'S TEST PYRAMID




Large
Medium
Small
DB


Large
Medium
Small
DB

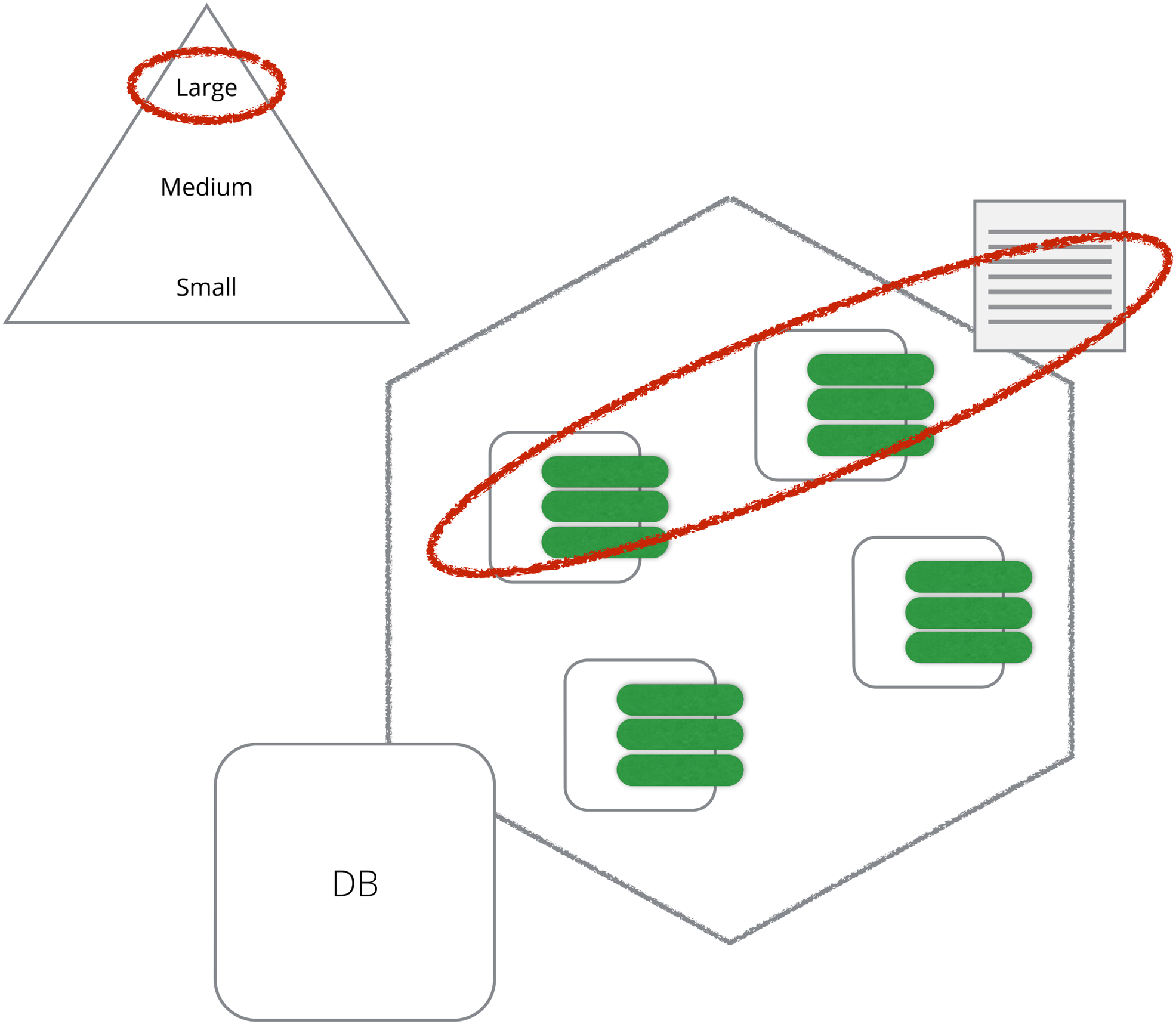

Large
Medium
Small
DB

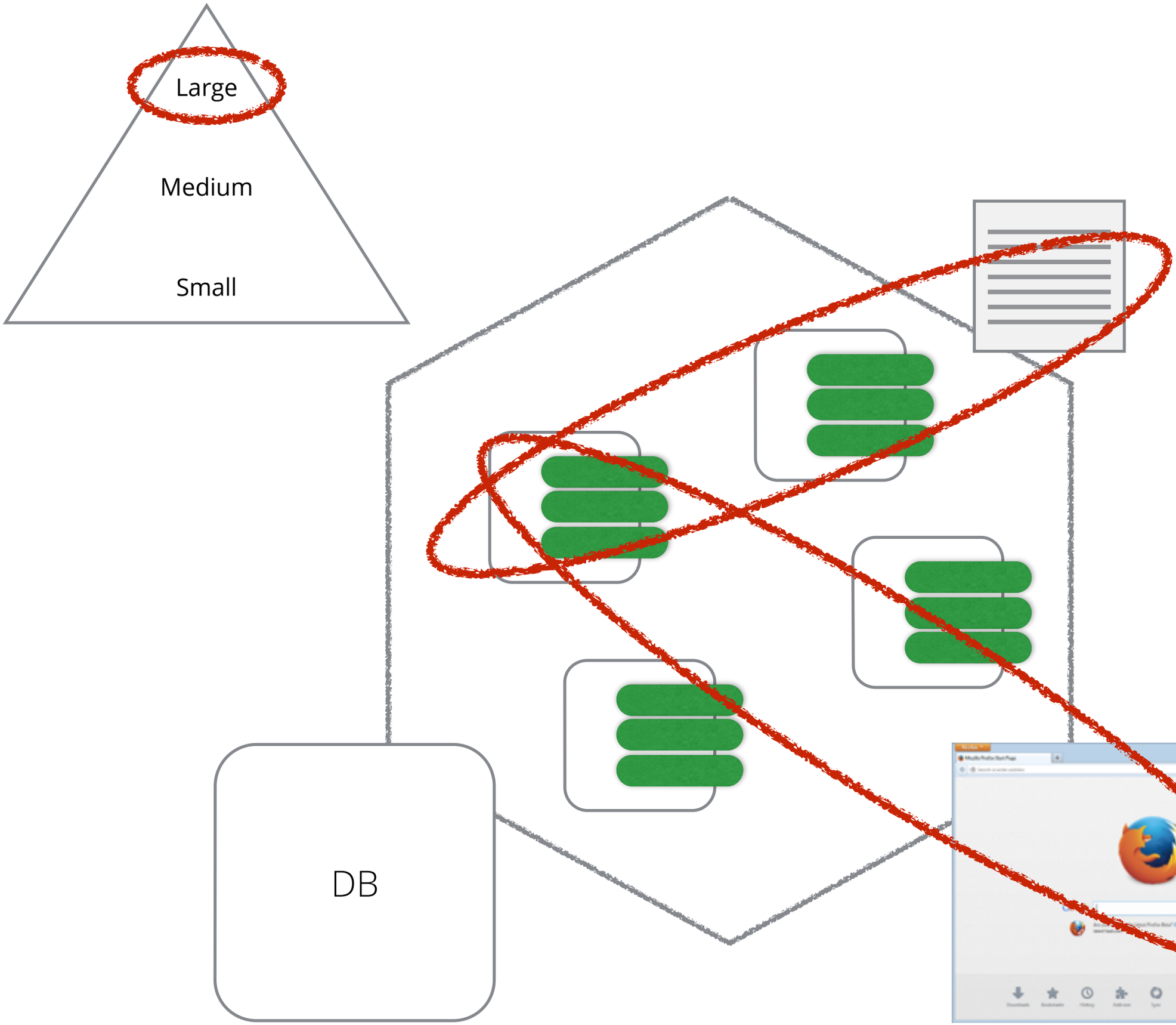


#geecon

@samnewman

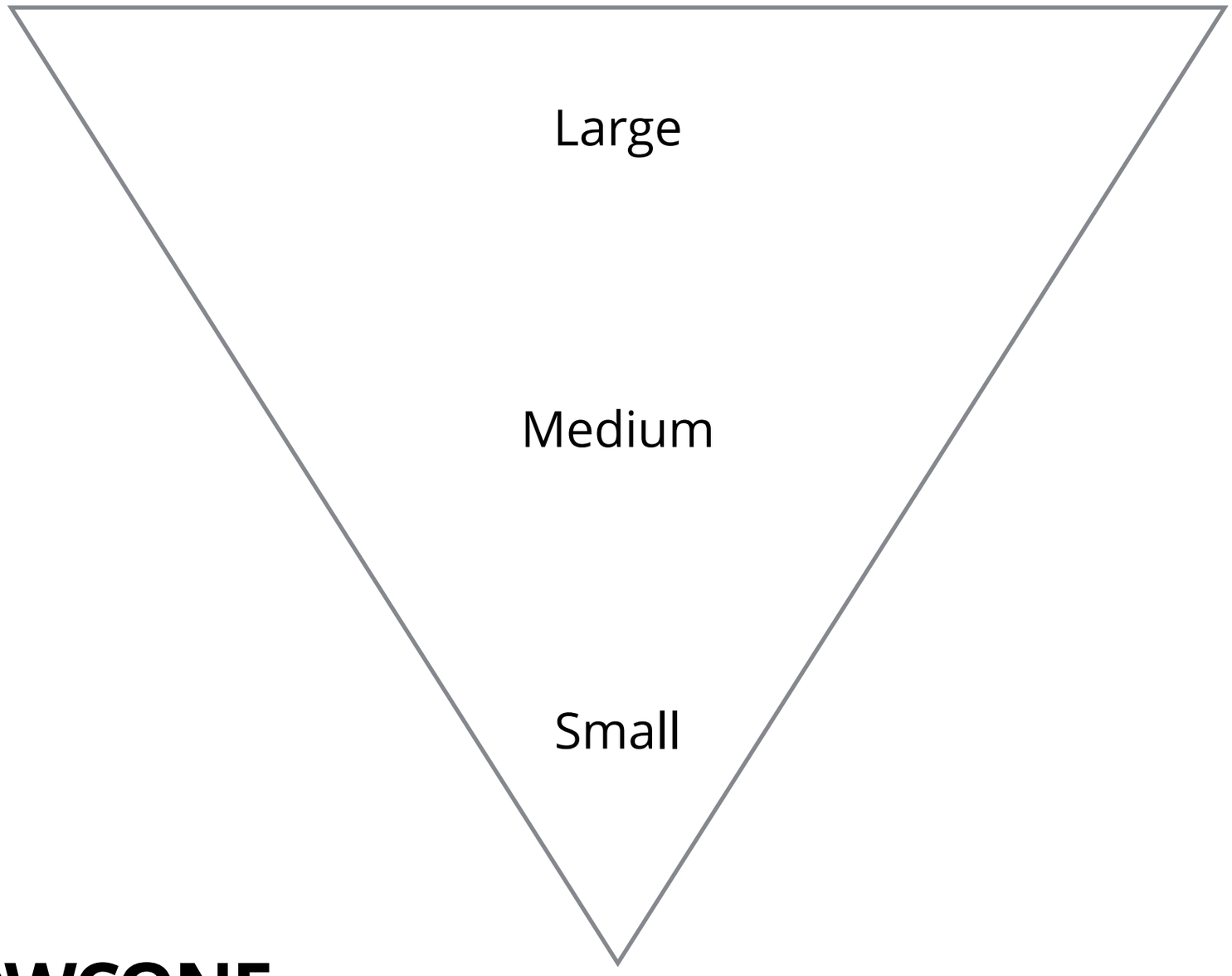

Large
Medium
Small
DB


Large
Medium
Small
DB

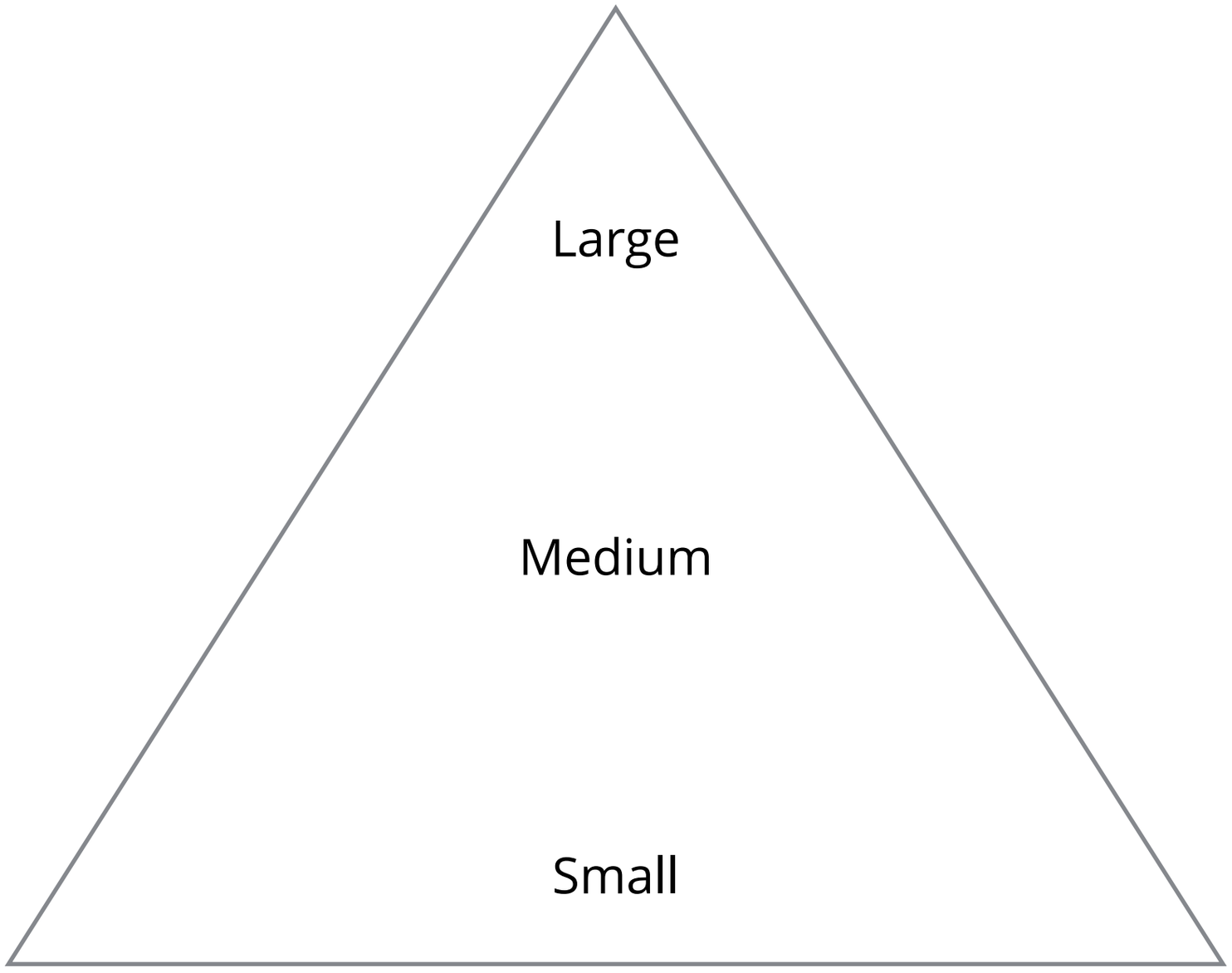


#geecon

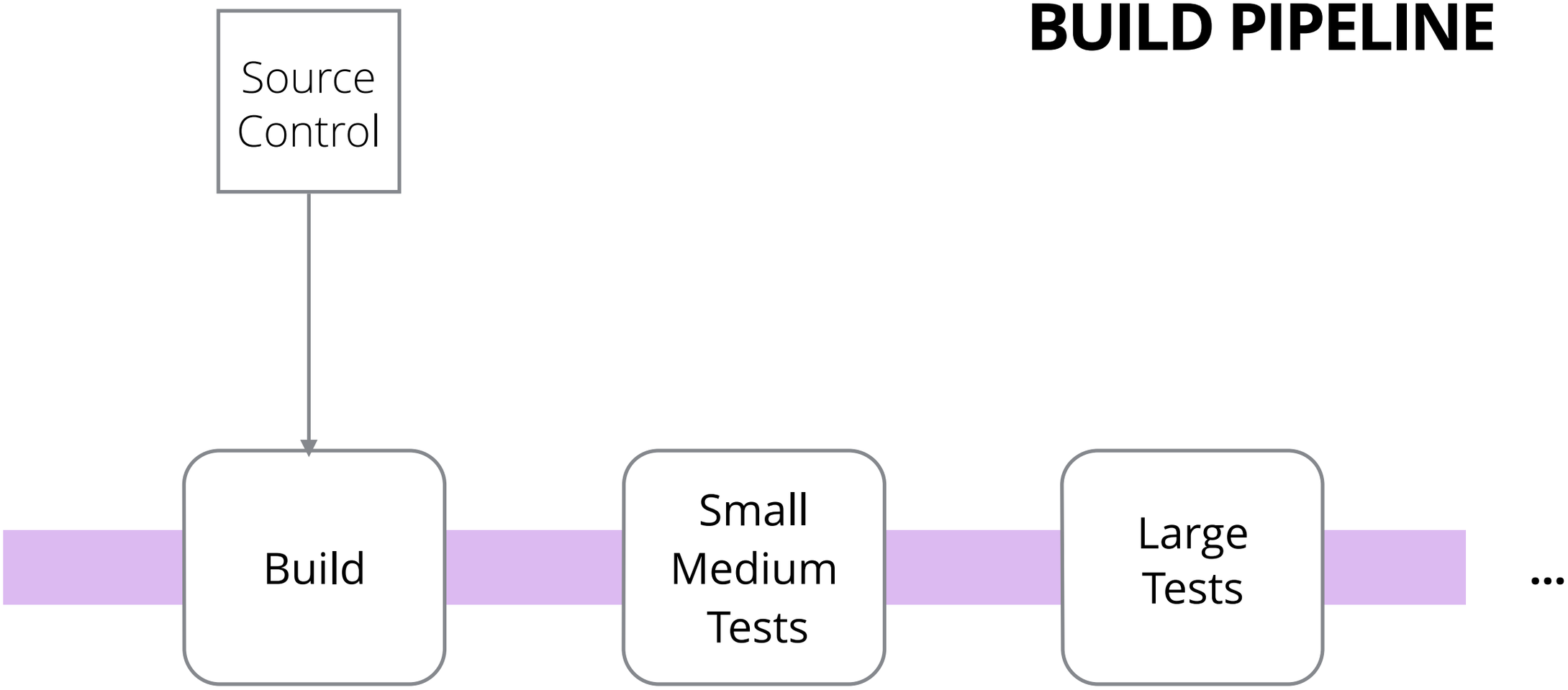
@samnewman



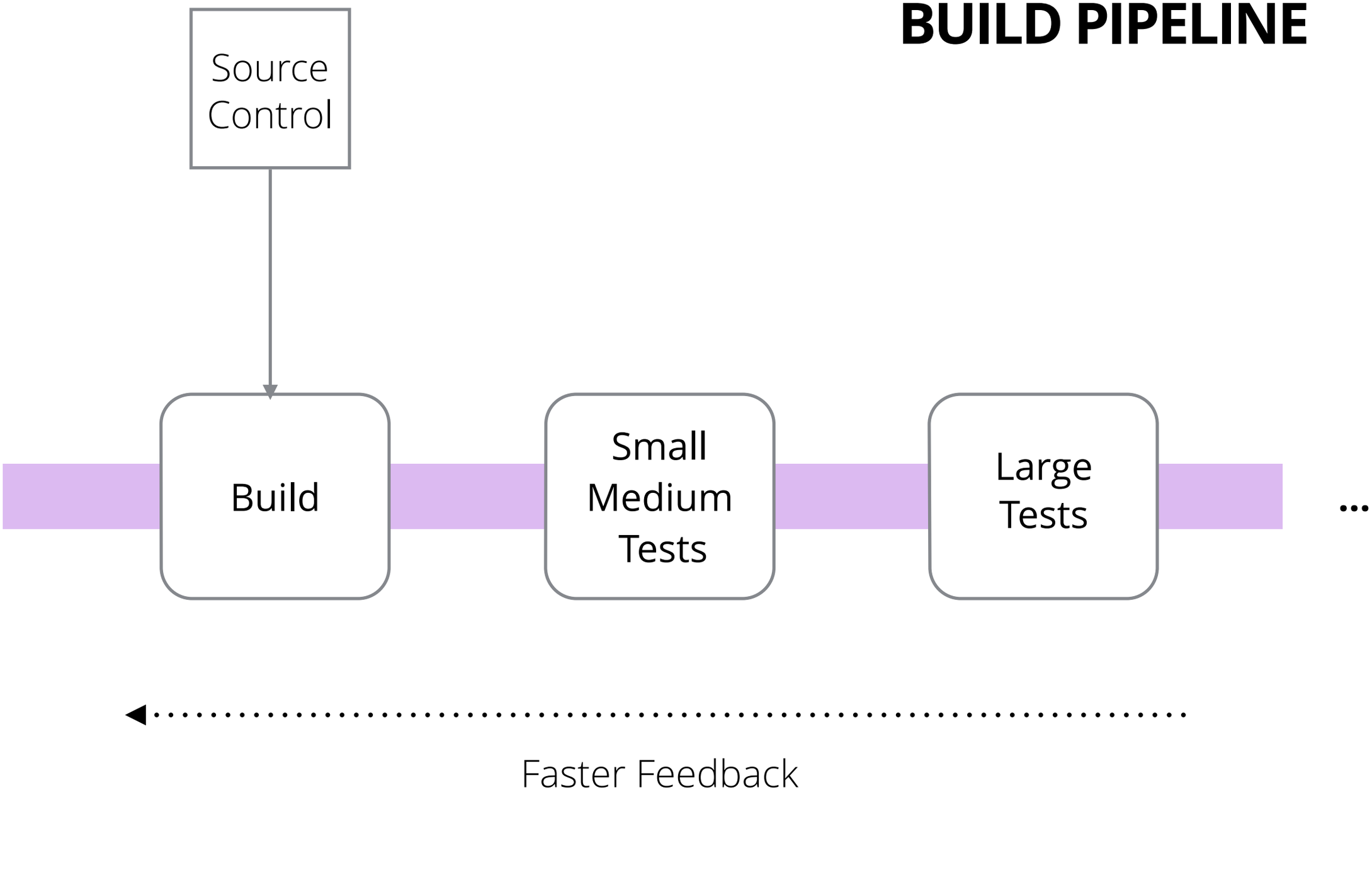
TEST SNOWCONE


Large
Medium
Small

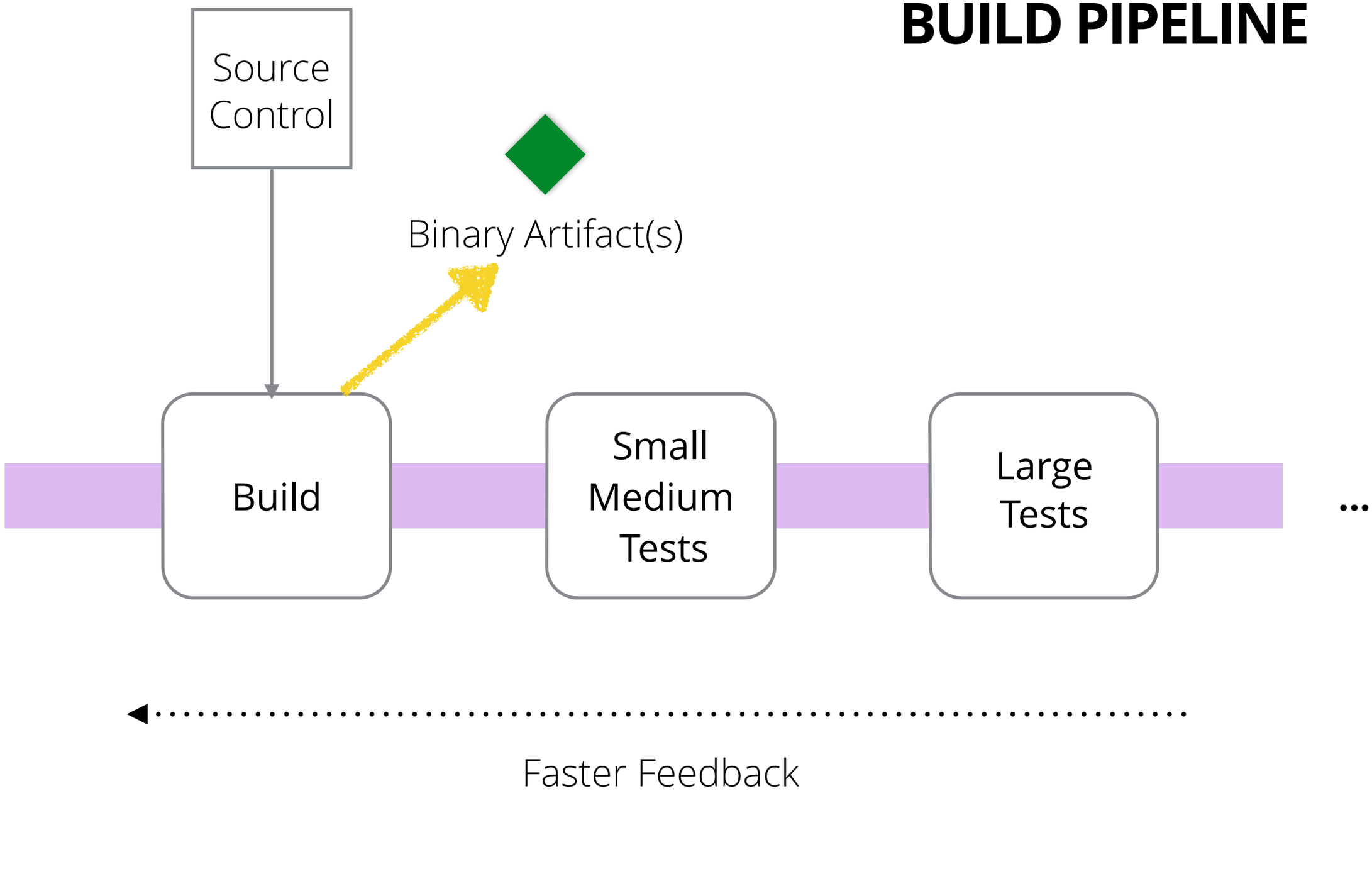
BUILD PIPELINE



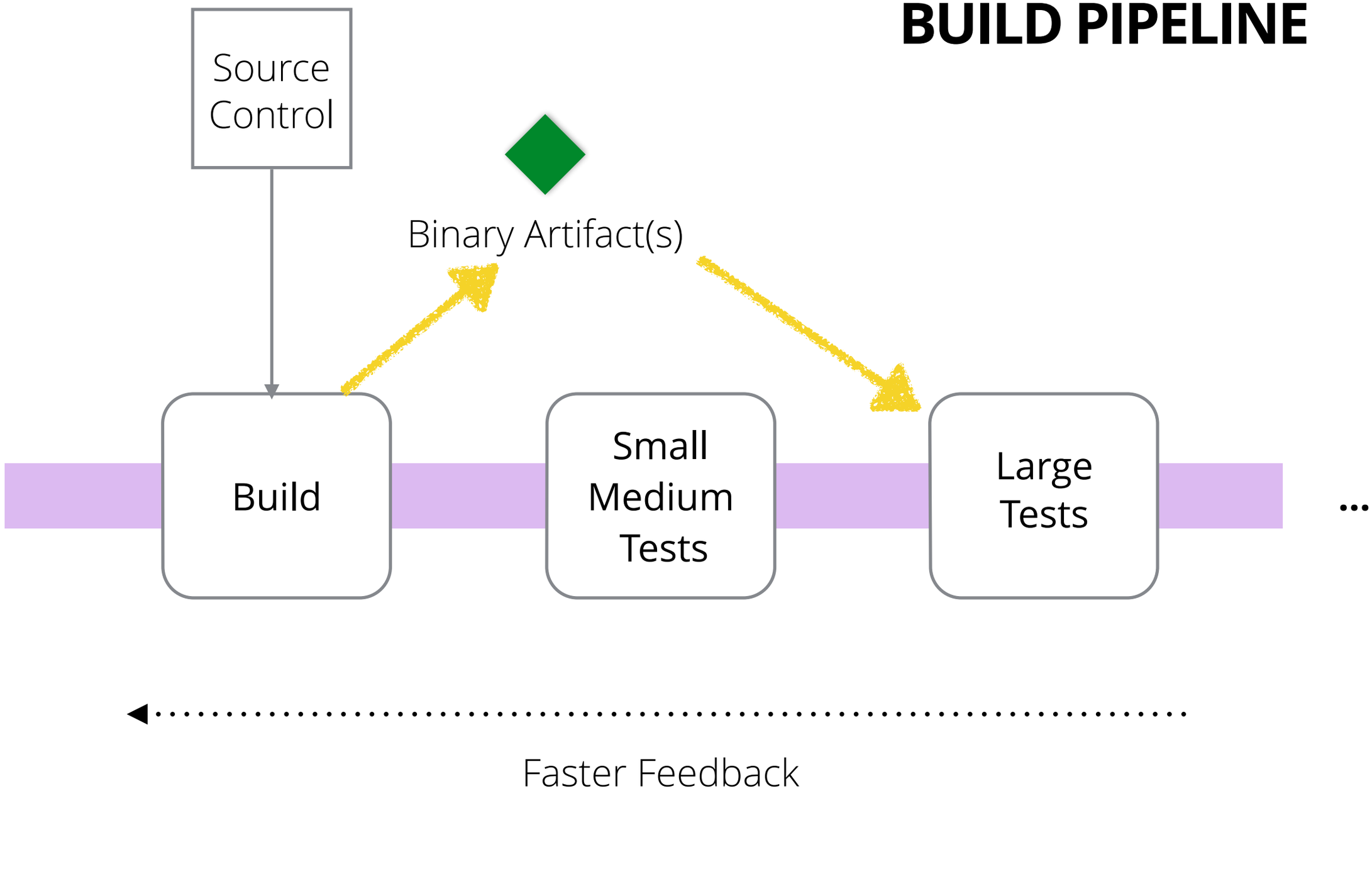
BUILD PIPELINE

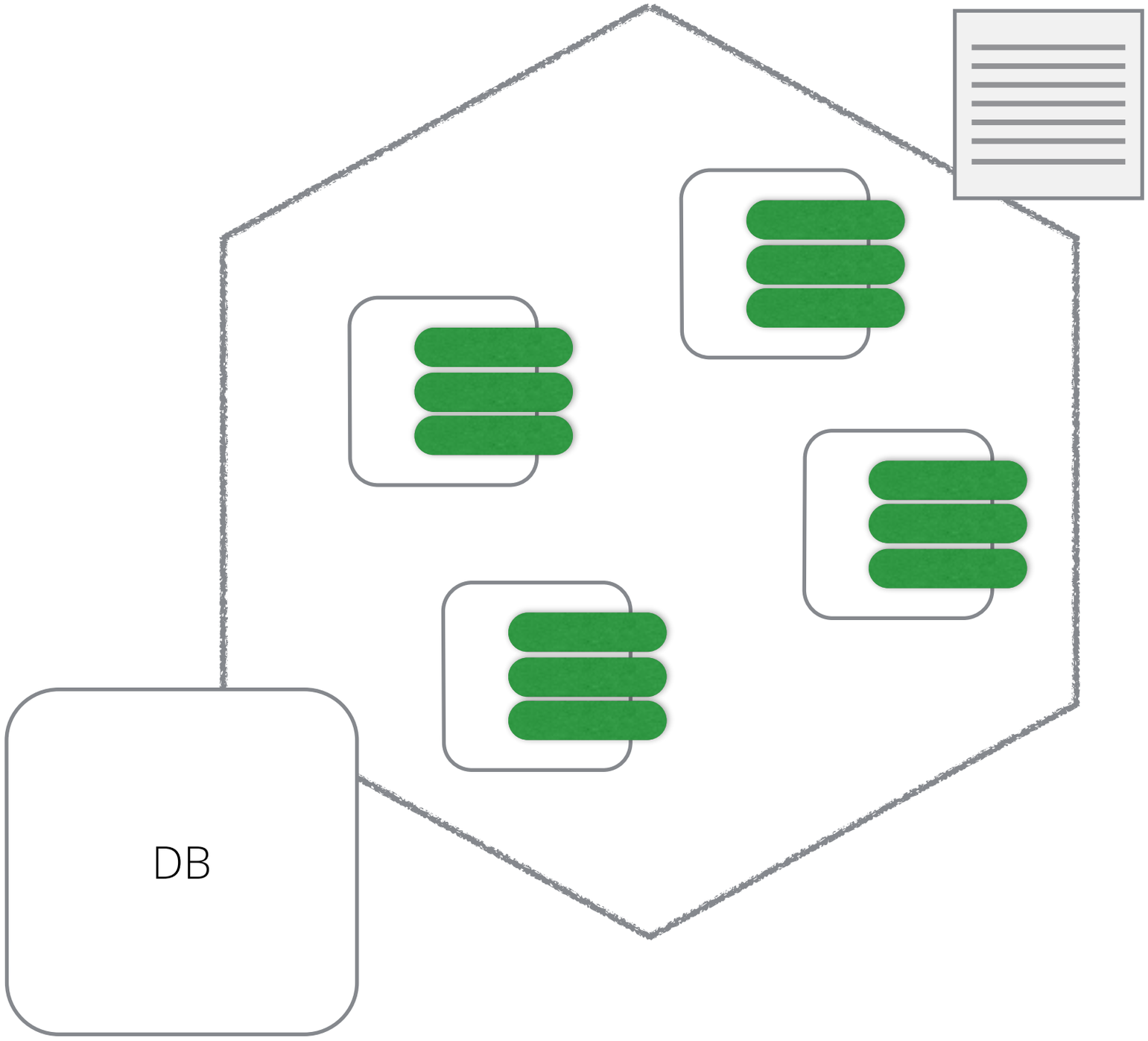


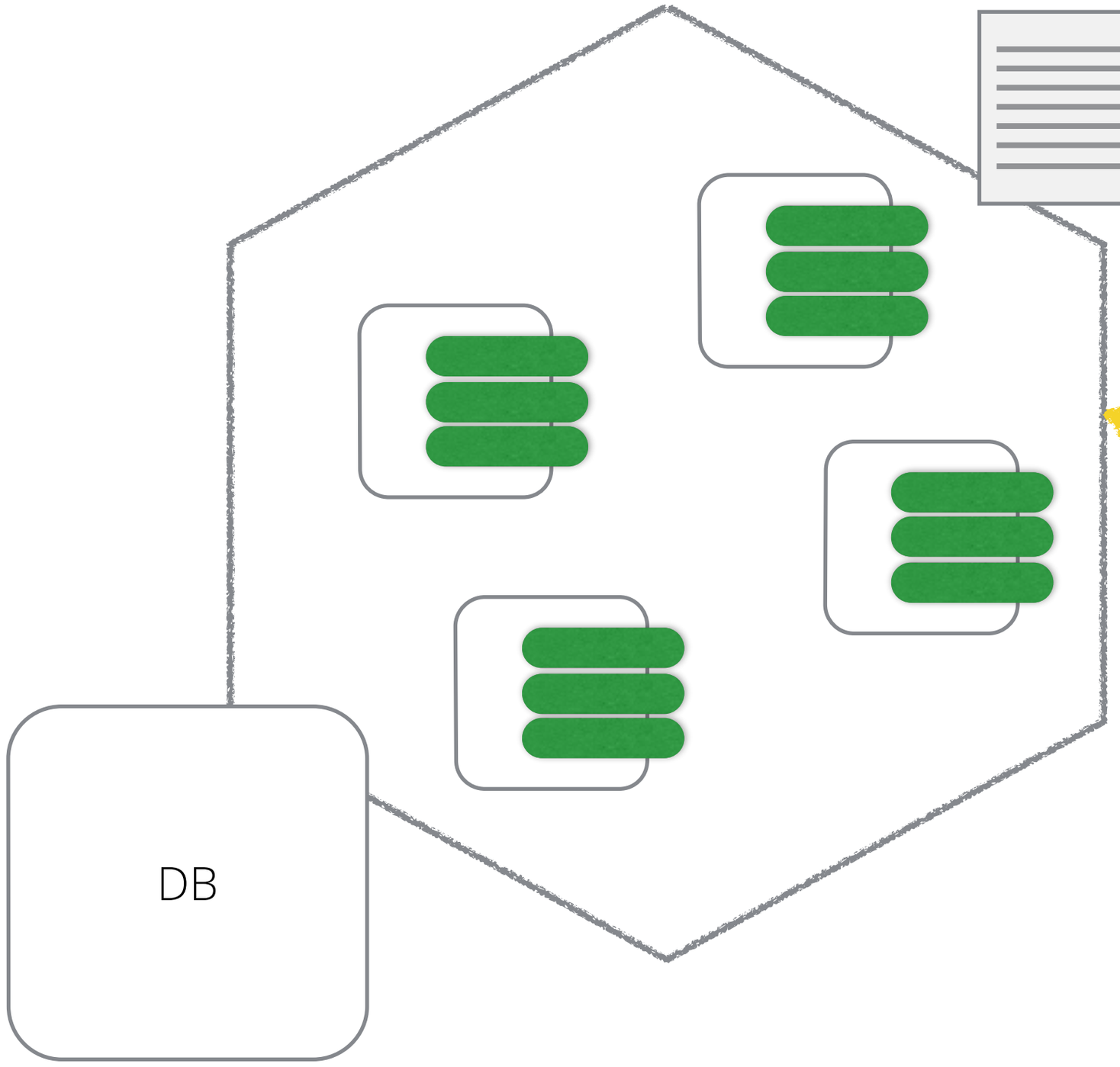
BUILD PIPELINE

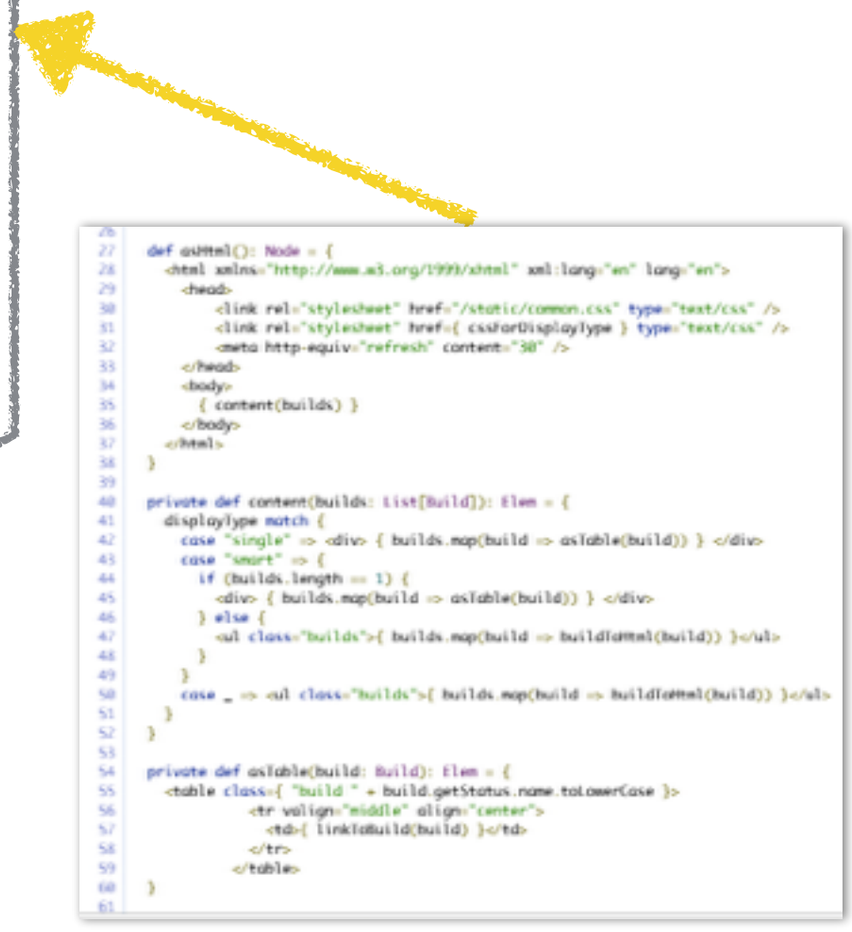


BUILD PIPELINE

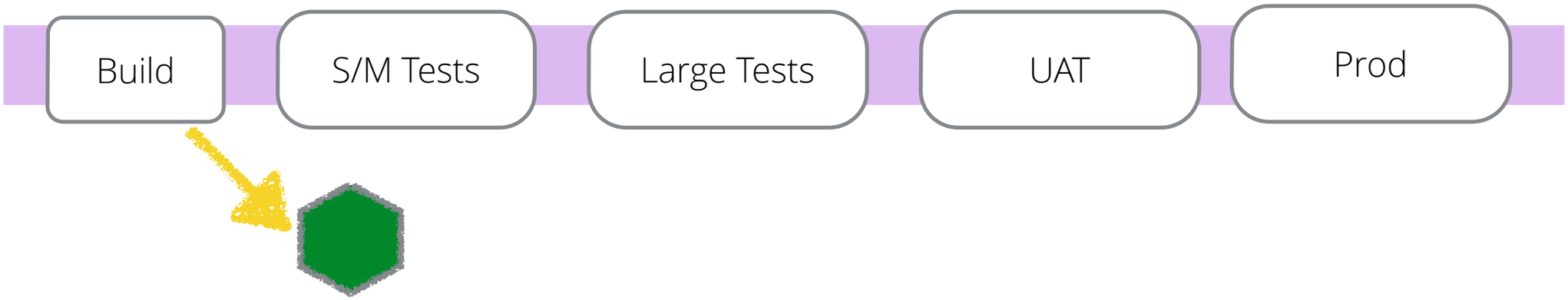


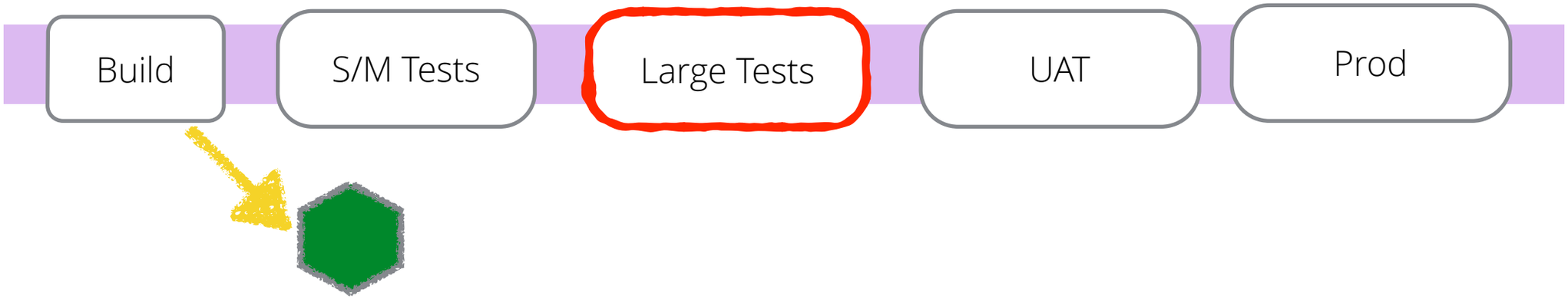

DB

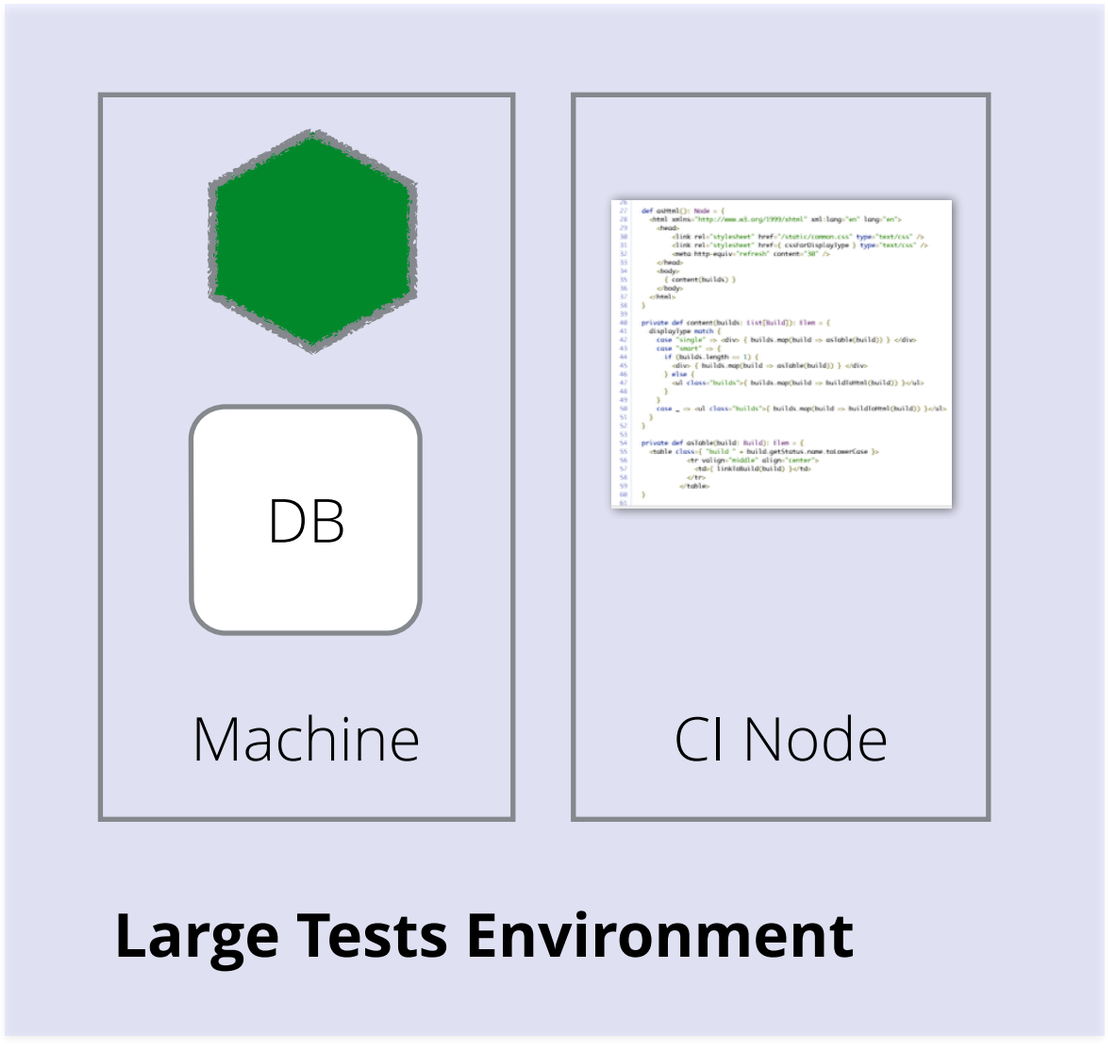




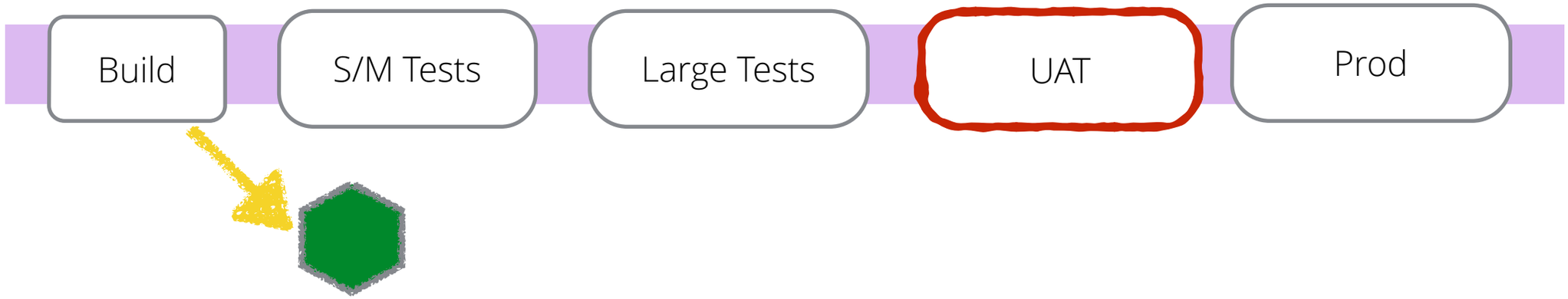
```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
            <tr valign="middle" align="center">
              <td>{ linkToBuild(build) }</td>
            </tr>
          </table>
}
```

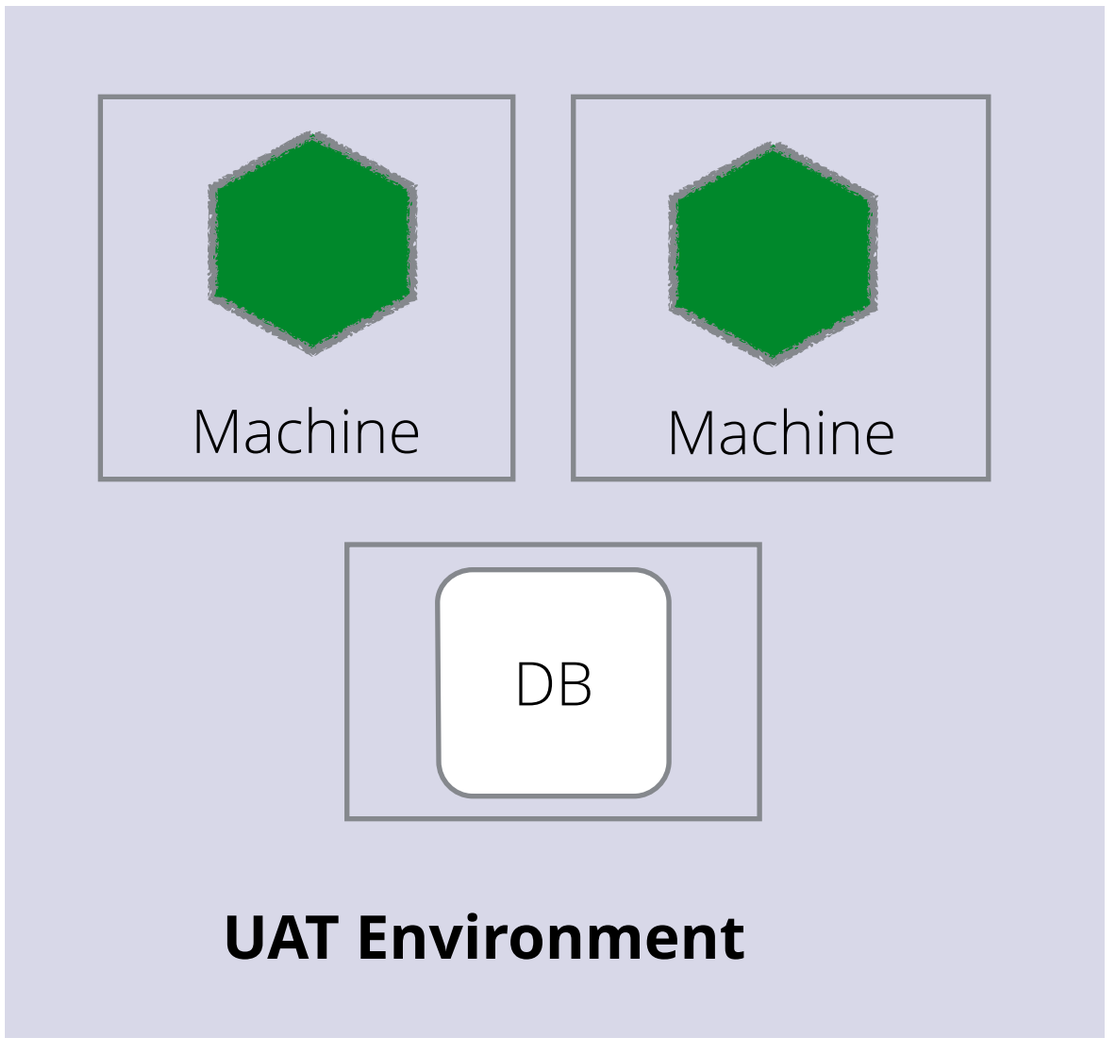

Build
S/M Tests
Large Tests
UAT
Prod

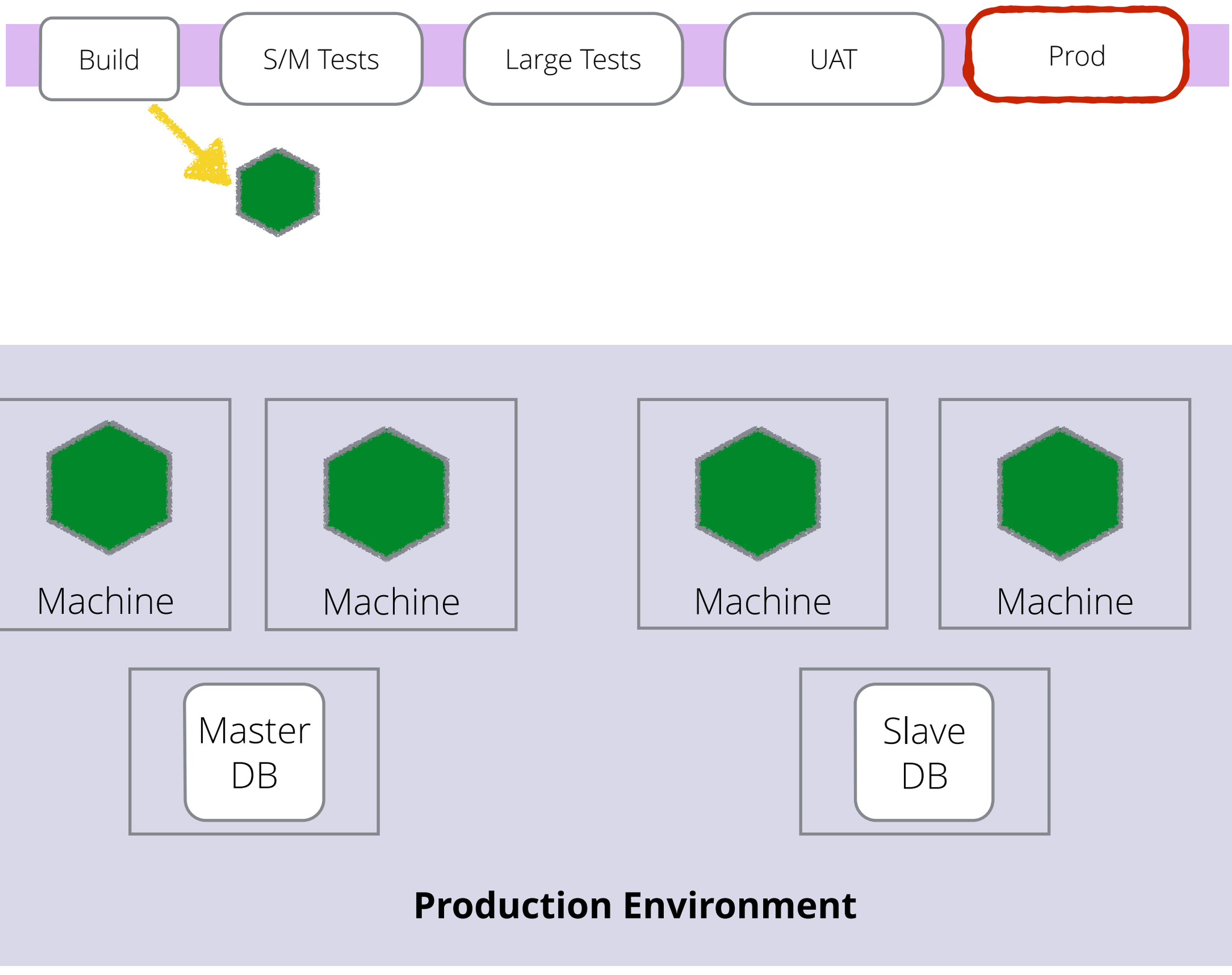


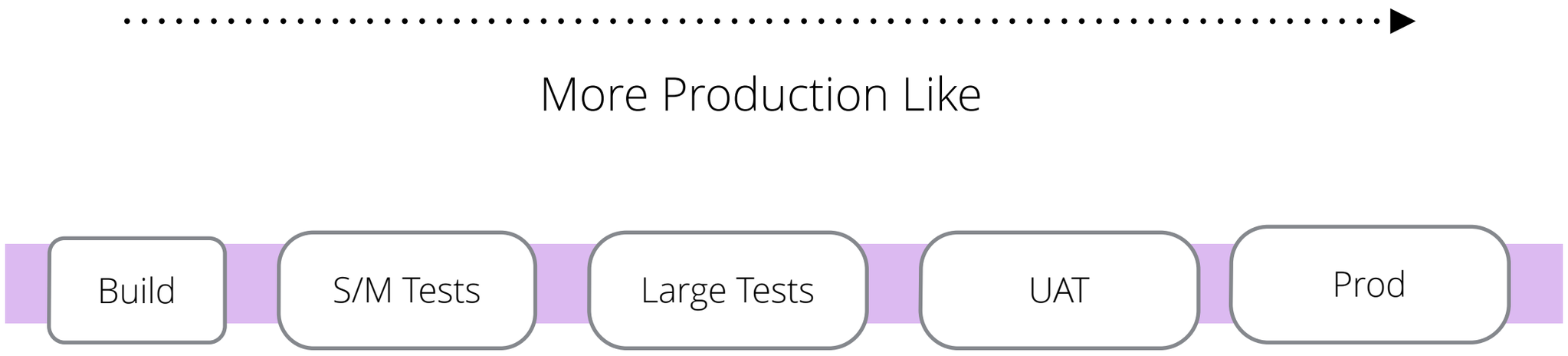


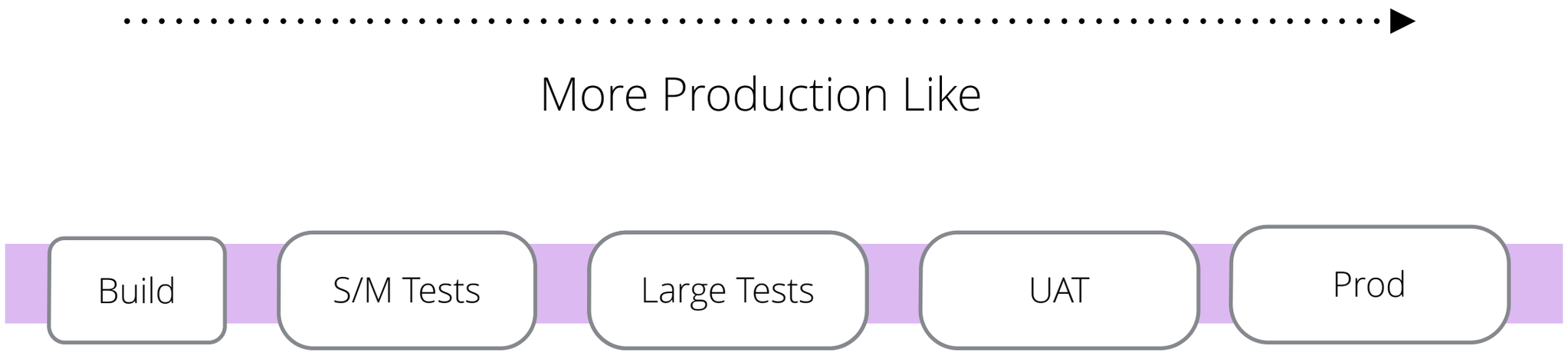
Large Tests Environment


Build
S/M Tests
Large Tests
UAT
Prod


Machine
Machine
DB
UAT Environment


Build
S/M Tests
Large Tests
UAT
Prod
Machine
Machine
Machine
Machine
Master DB
Slave DB
Production Environment


More Production Like
Build
S/M Tests
Large Tests
UAT
Prod


More Production Like
Build
S/M Tests
Large Tests
UAT
Prod

More Production Like

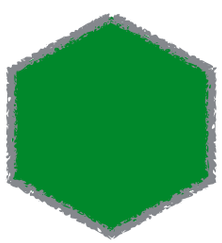
Faster Feedback


Customer Service

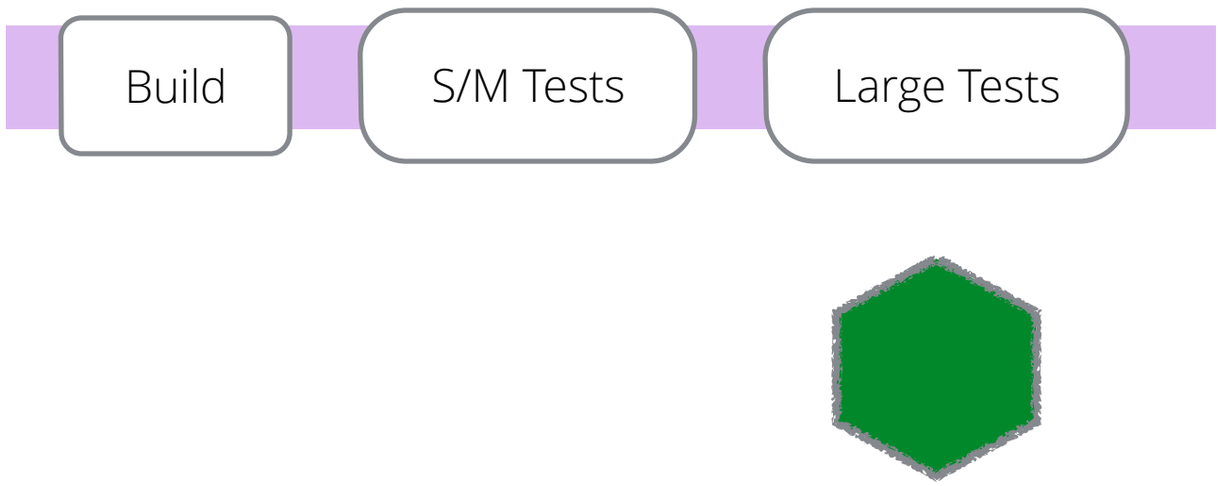

Customer Service

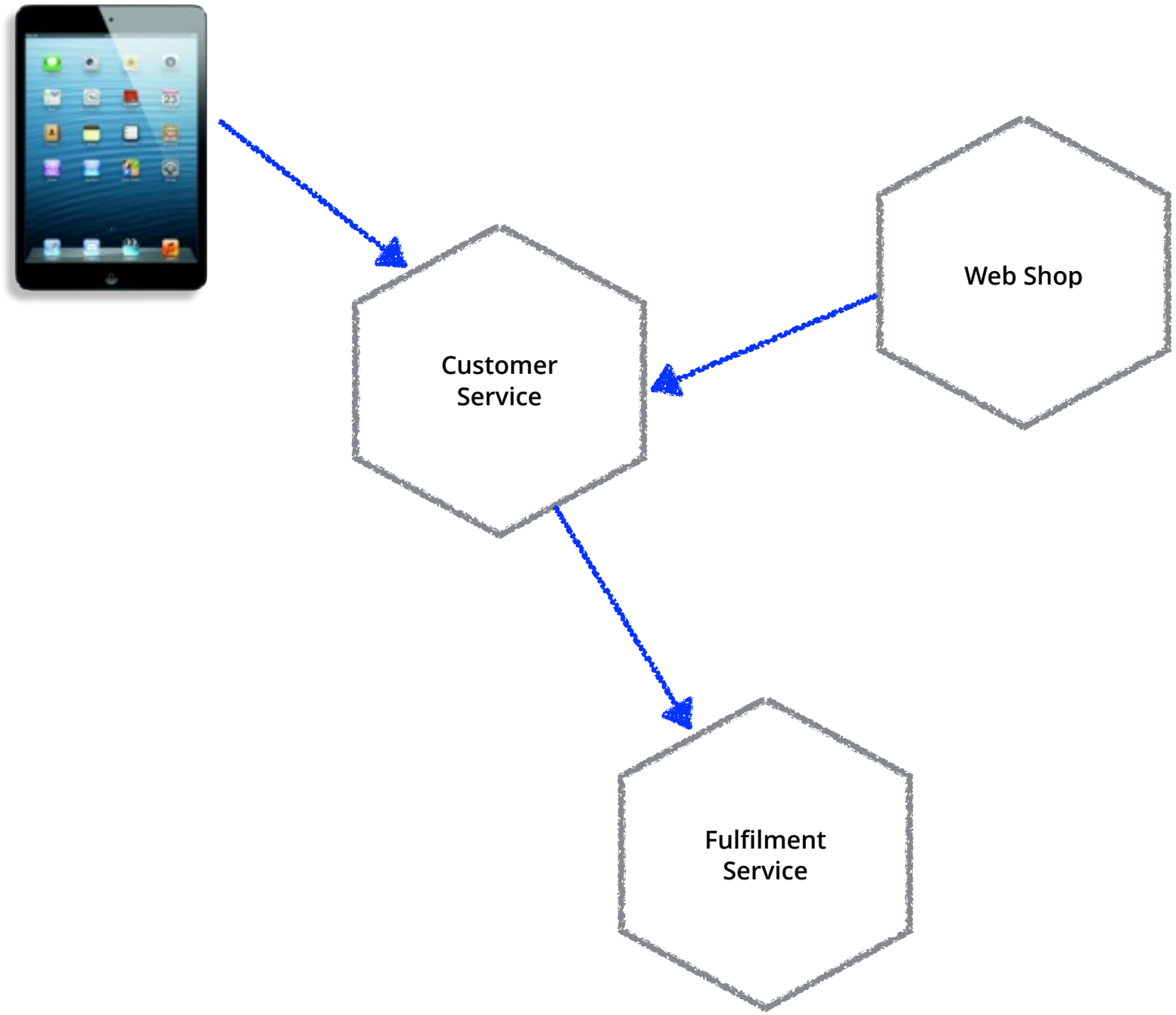

Build
S/M Tests
Large Tests

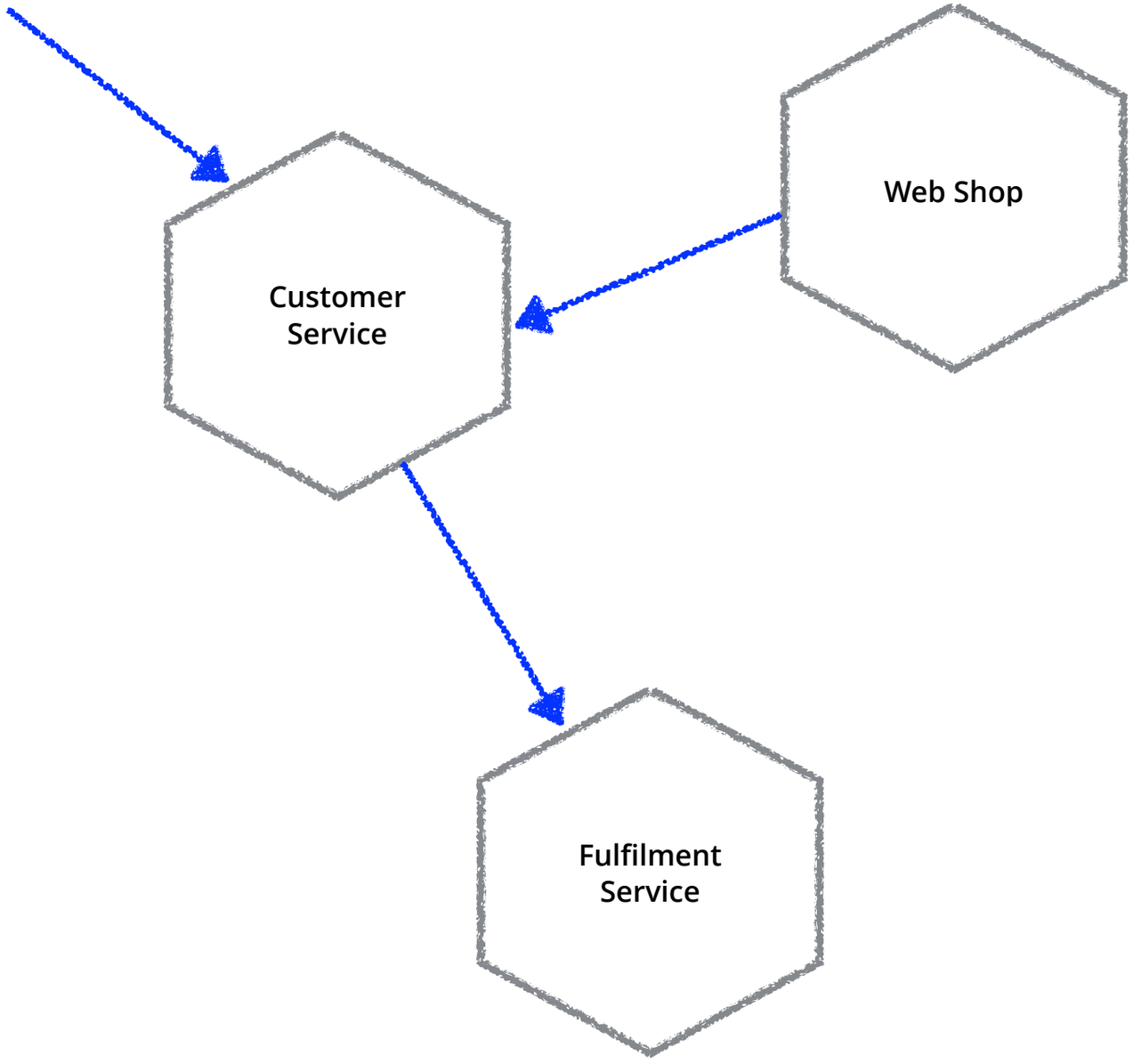
Customer Service

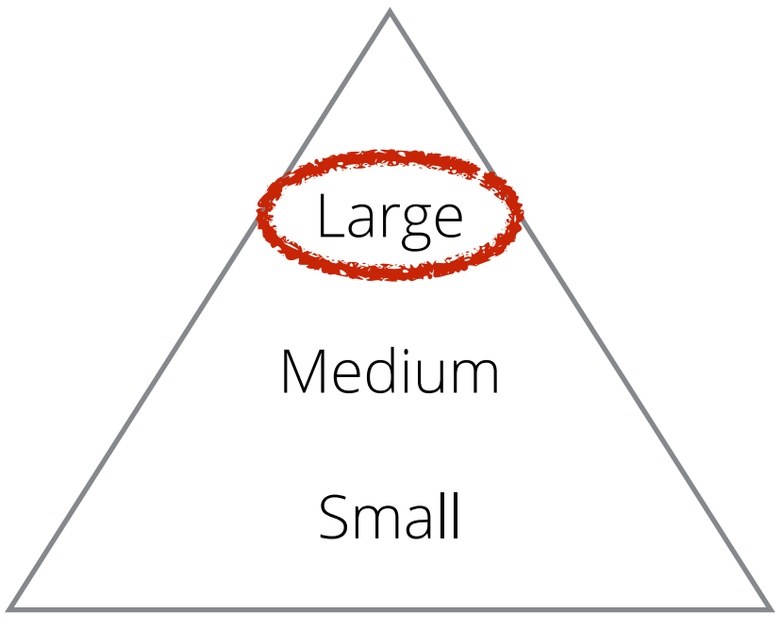


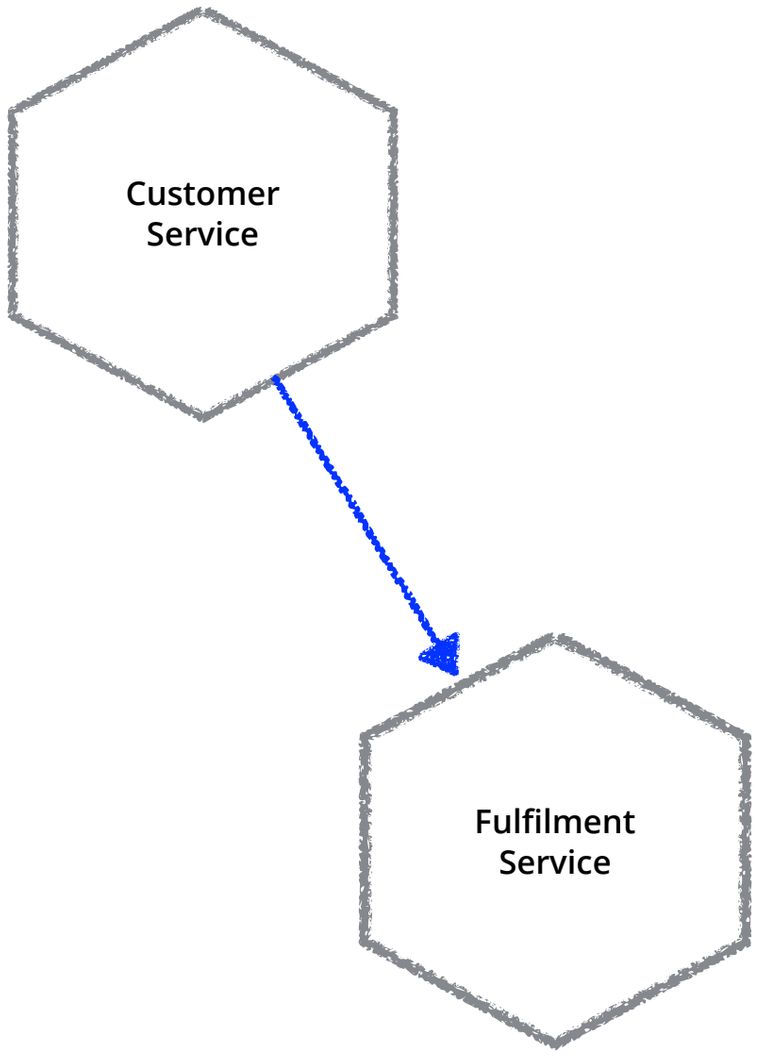

Customer Service

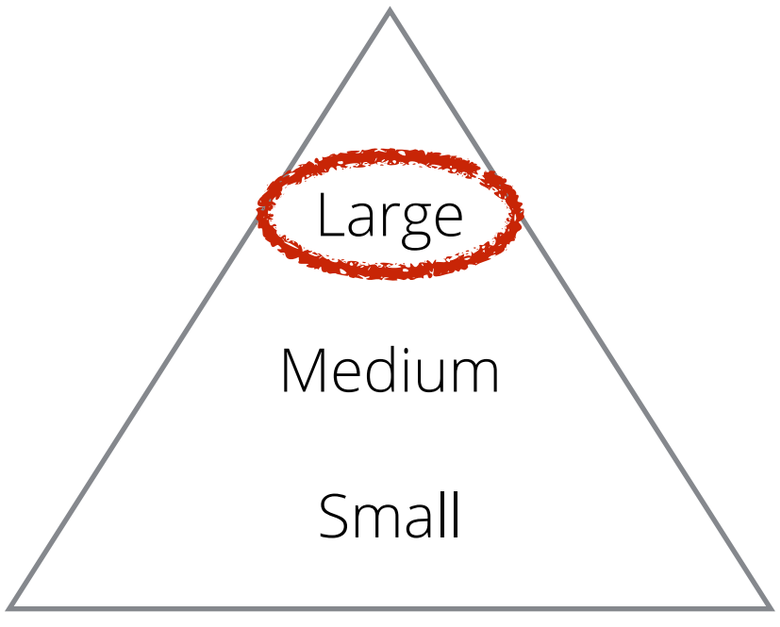

Build
S/M Tests
Large Tests

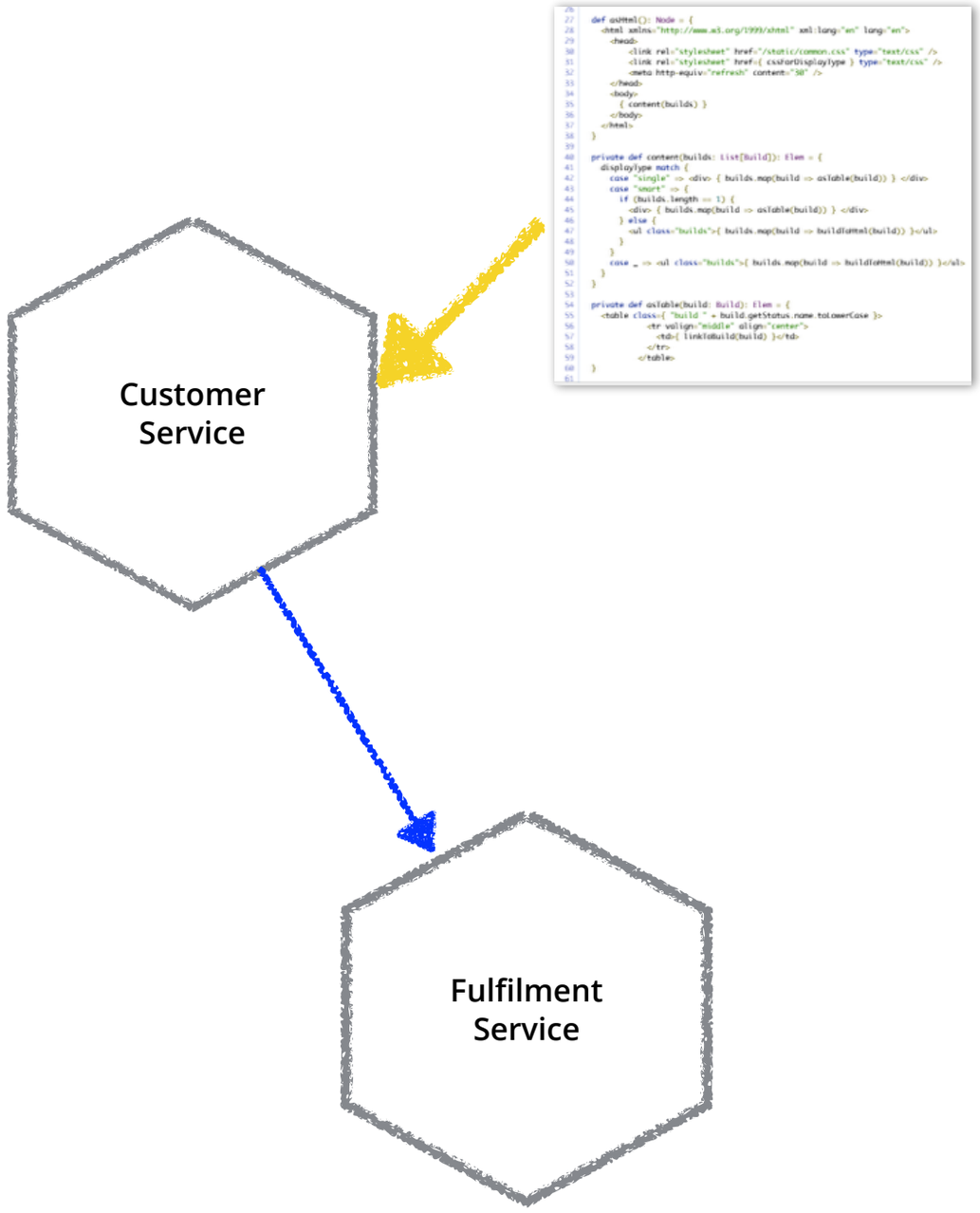

Customer
Service
Web Shop
Fulfilment
Service

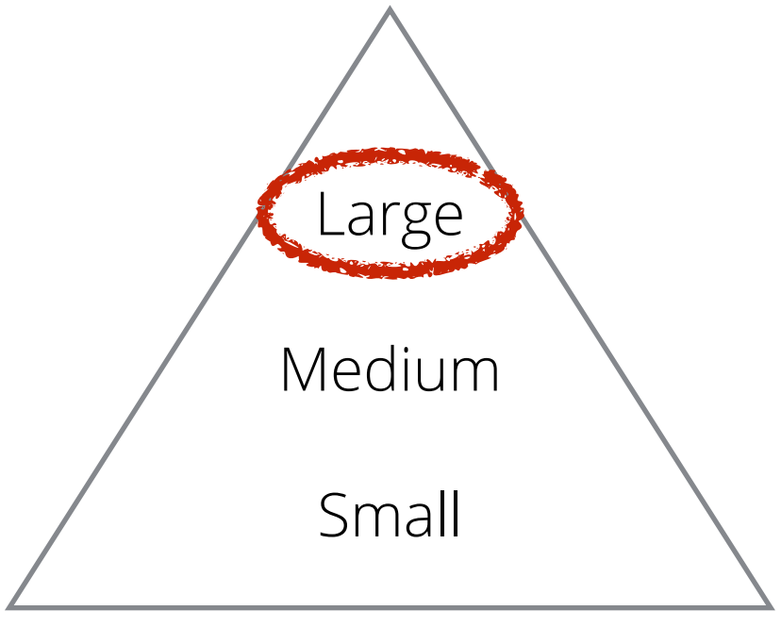

Customer Service
Web Shop
Fulfilment Service

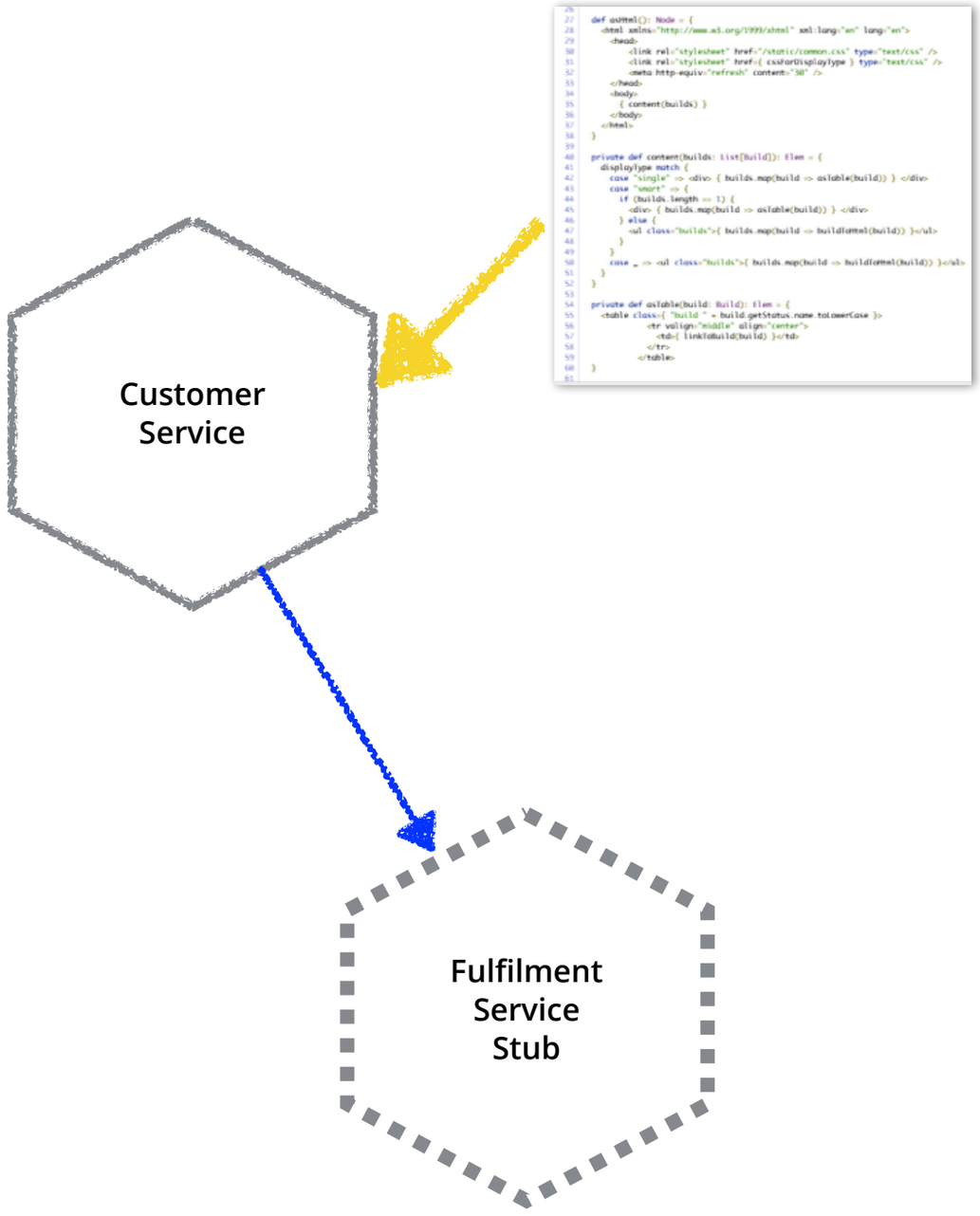

Large
Medium
Small

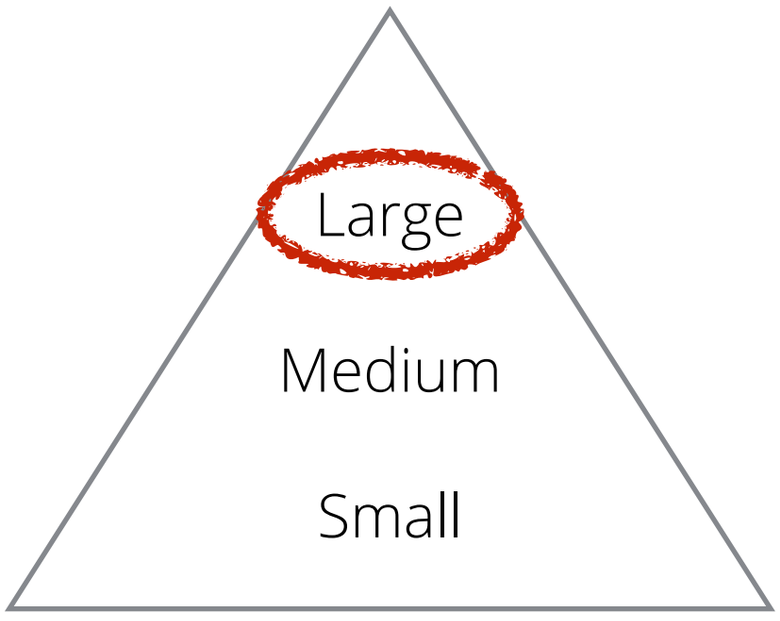

Customer
Service
Fulfilment
Service


Large
Medium
Small


Customer Service
Fulfilment Service


Large
Medium
Small


Customer Service
Fulfilment Service Stub


Large
Medium
Small

mountebank - over the wire test doubles

Fork me on GitHub

home imposters logs config

the apothecary

getting started
install options
command line
faq
support
glossary

api:

overview
mock verification
stubs
stub predicates
stub proxies
stub injection
errors

protocols:

Welcome, friend

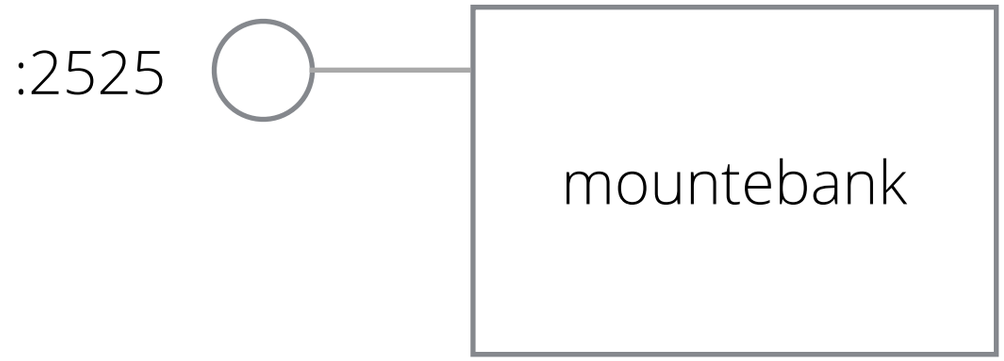
. Simply point
uld with

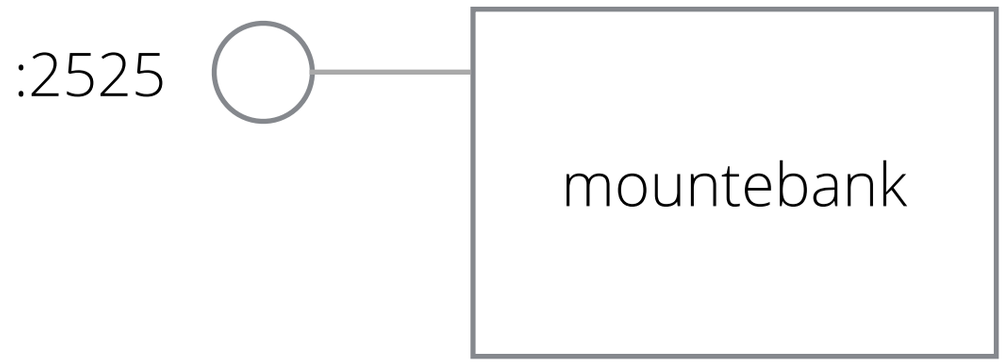
mountebank employs a legion of *imposters* to act as on-demand test doubles. Your test communicates to mountebank over http using the api to set up stubs, record and replay proxies, and verify mock expectations. In the typical use case, each test will start an imposter during test setup and stop an imposter during test teardown.

mountebank employs several types of imposters, each responding to a specific protocol. Typically, your test will tell the imposter which port to bind to, and the imposter will open the corresponding socket.

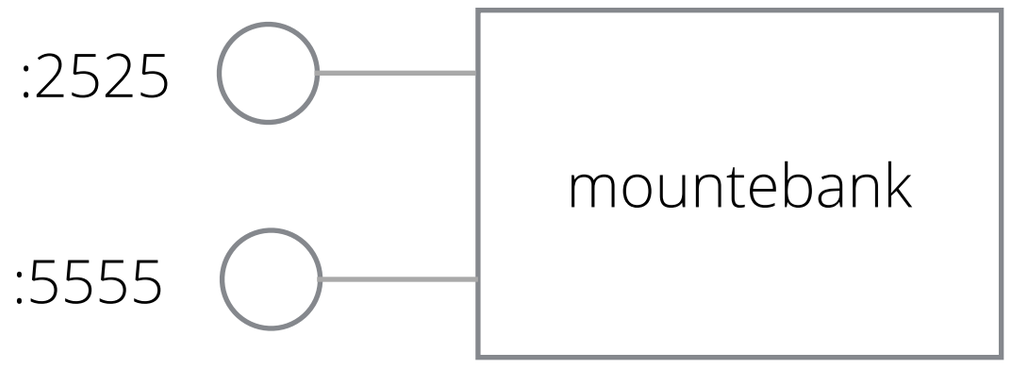
Mountebank

http://www.mbtest.org

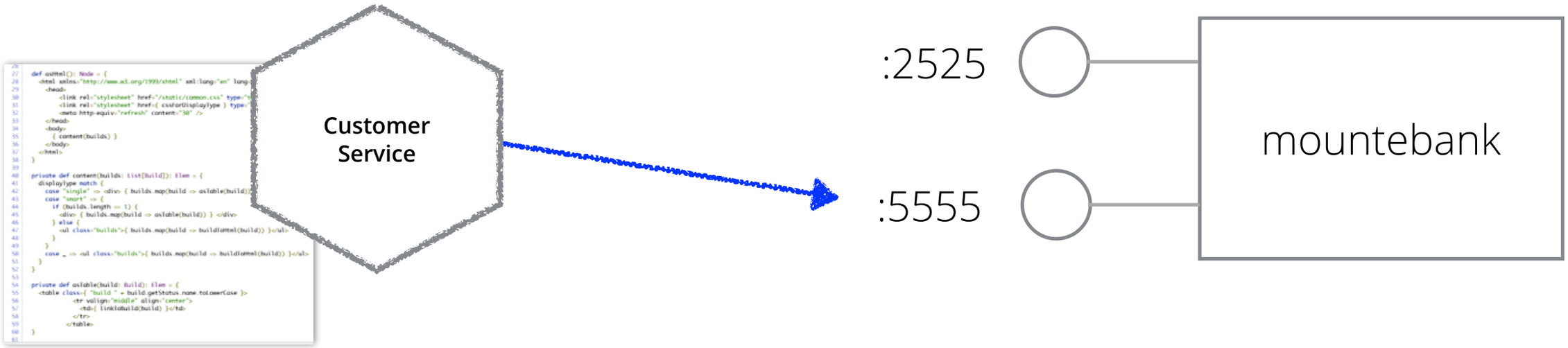

:2525
mountebank




```
cat << EOF > imposter.json
{
  "port": 5555,
  "protocol": "tcp",
  "mode": "binary",
  "stubs": [{
    "responses": [
      { "is": { "data": "aGVsbG8sIHdvcmxkIQ==" }}
    ],
    "predicates": [{ "contains": { "data": "c2F5SGVsbG8=" } }]
  }]
}
EOF

curl -i -X POST -H 'Content-Type: application/json' -d@imposter.json http://localhost:2525/imposters
```

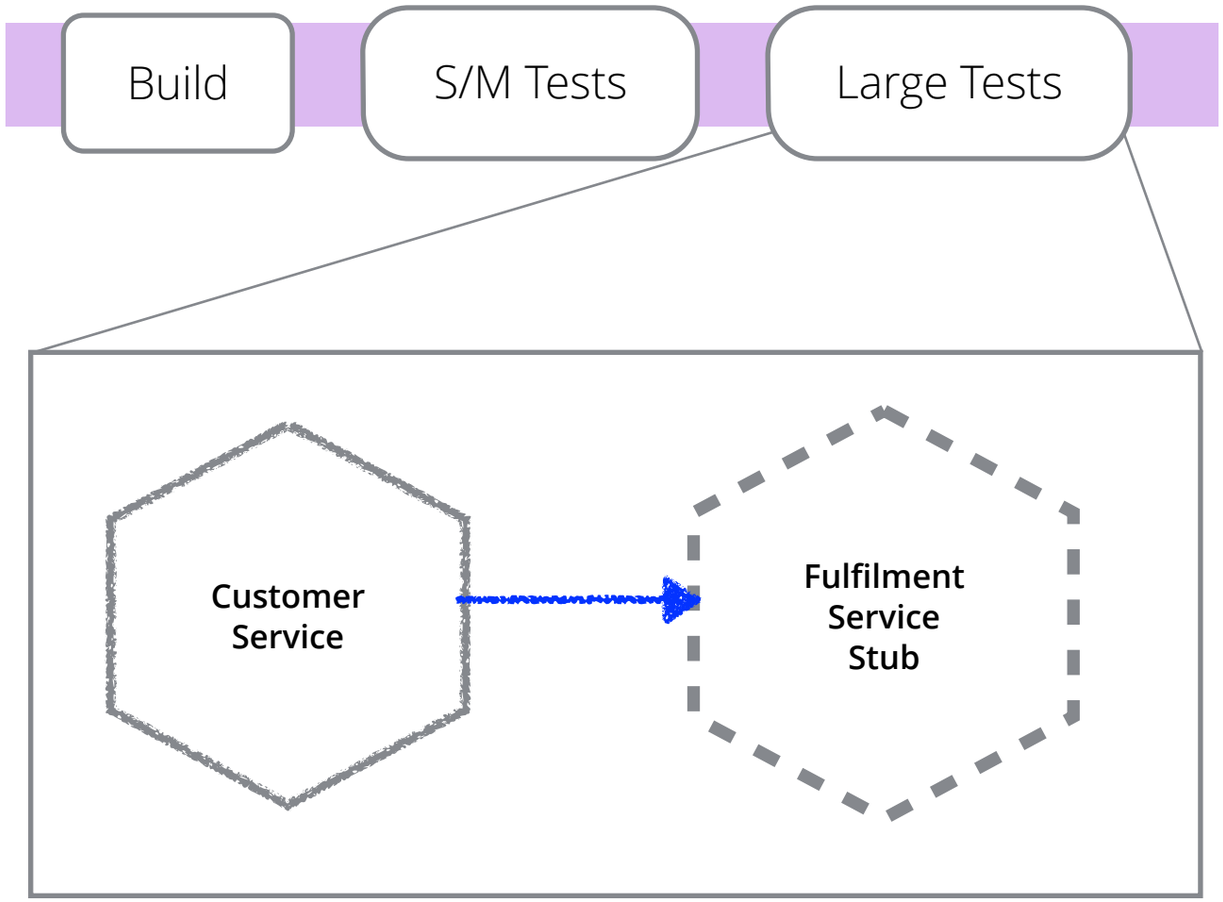


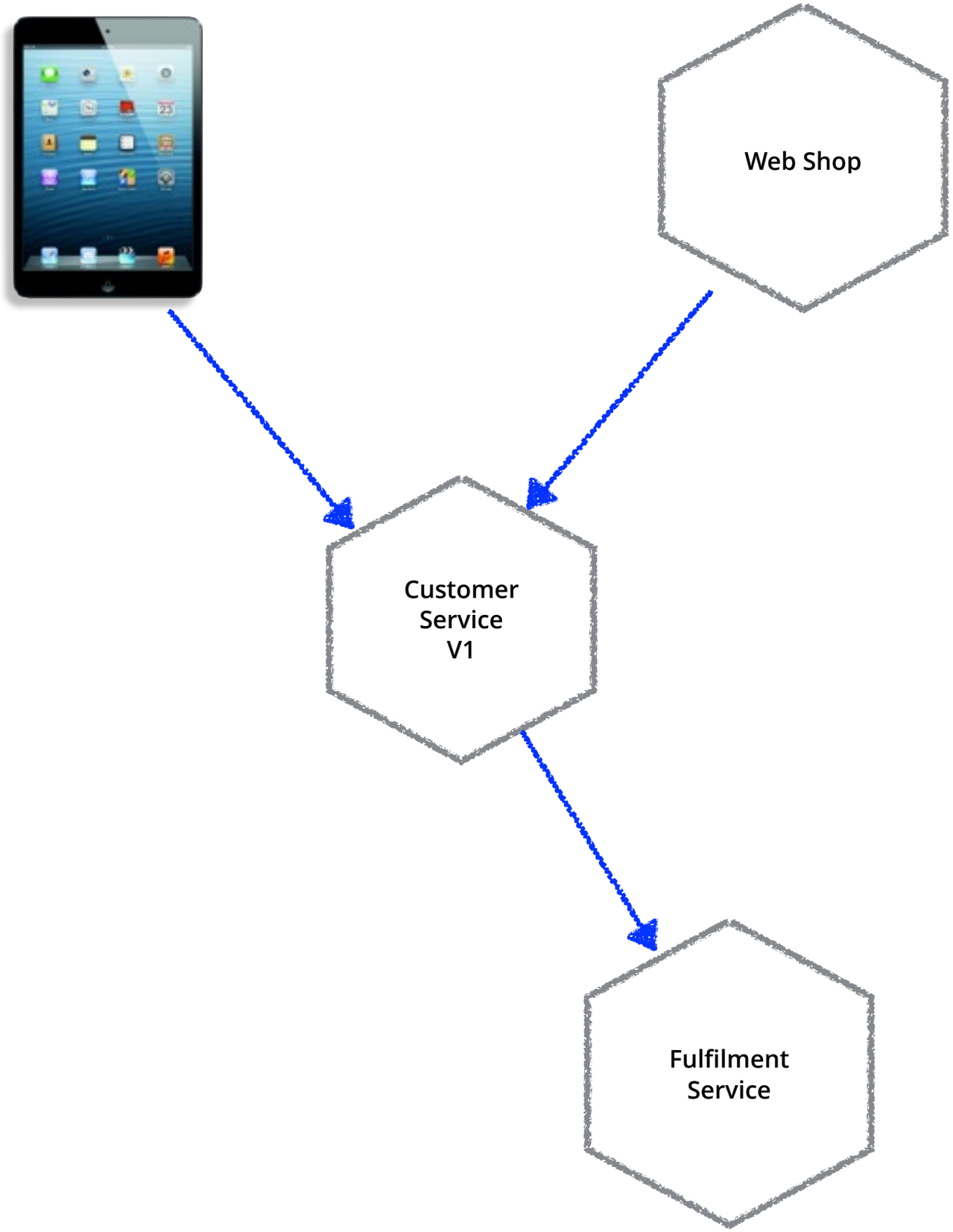

```
cat << EOF > imposter.json
{
  "port": 5555,
  "protocol": "tcp",
  "mode": "binary",
  "stubs": [{
    "responses": [
      { "is": { "data": "aGVsbG8sIHdvcmxkIQ==" }}
    ],
    "predicates": [{ "contains": { "data": "c2F5SGVsbG8=" } }]
  }]
}
EOF

curl -i -X POST -H 'Content-Type: application/json' -d@imposter.json http://localhost:2525/imposters
```

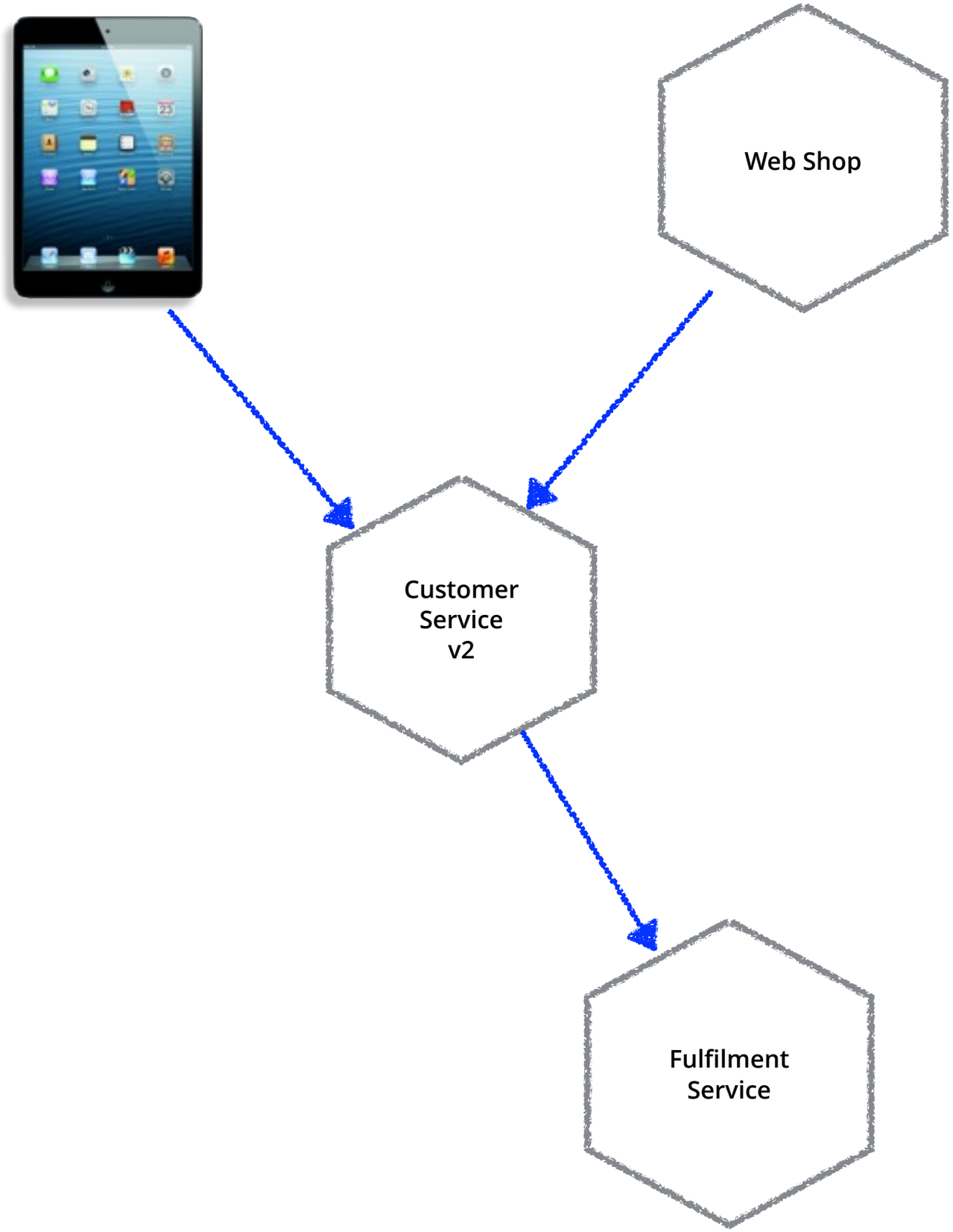


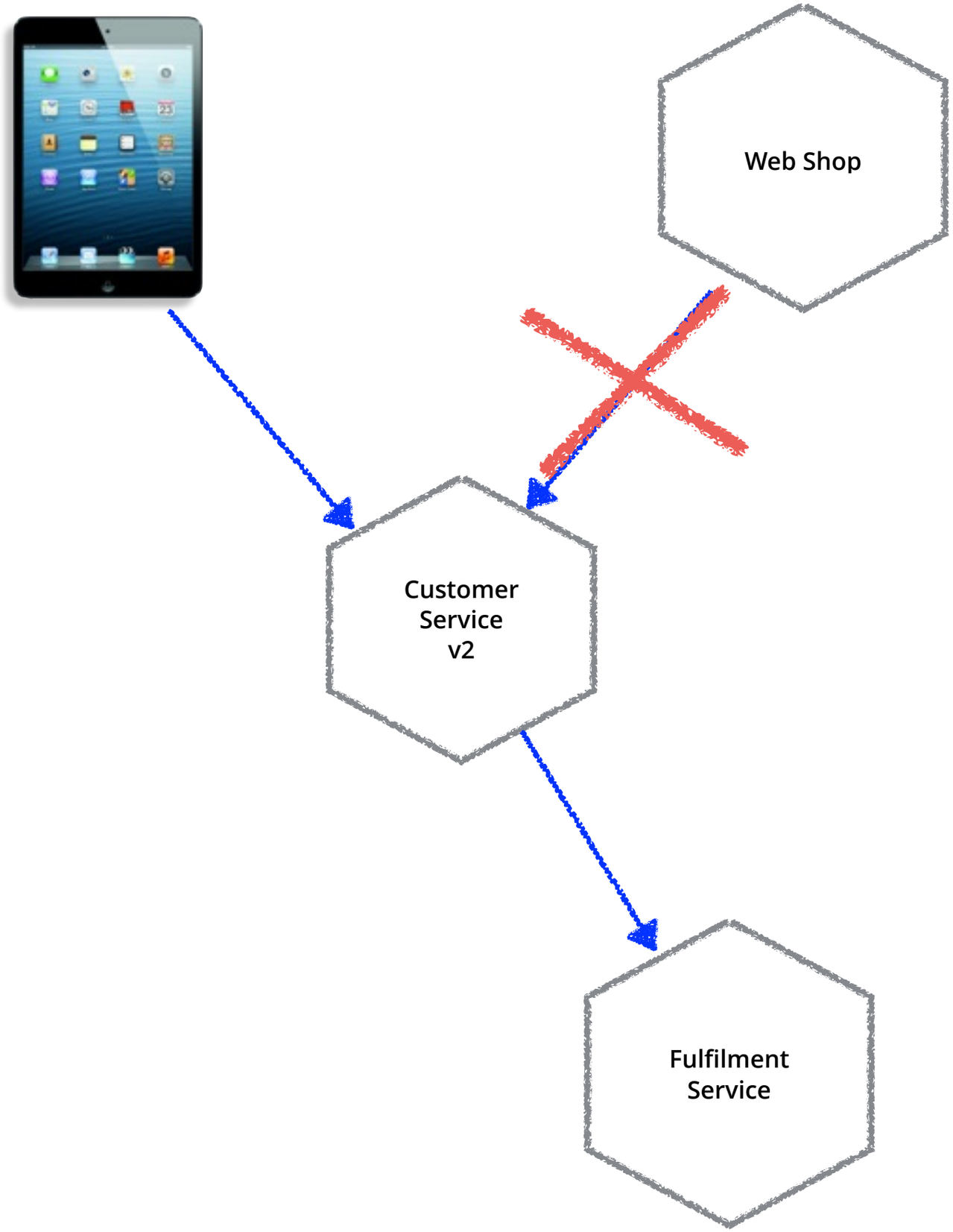

```
cat << EOF > imposter.json
{
  "port": 5555,
  "protocol": "tcp",
  "mode": "binary",
  "stubs": [{
    "responses": [
      { "is": { "data": "aGVsbG8sIHdvcmxkIQ==" }}
    ],
    "predicates": [{ "contains": { "data": "c2F5SGVsbG8=" } }]
  }]
}
EOF

curl -i -X POST -H 'Content-Type: application/json' -d@imposter.json http://localhost:2525/imposters
```

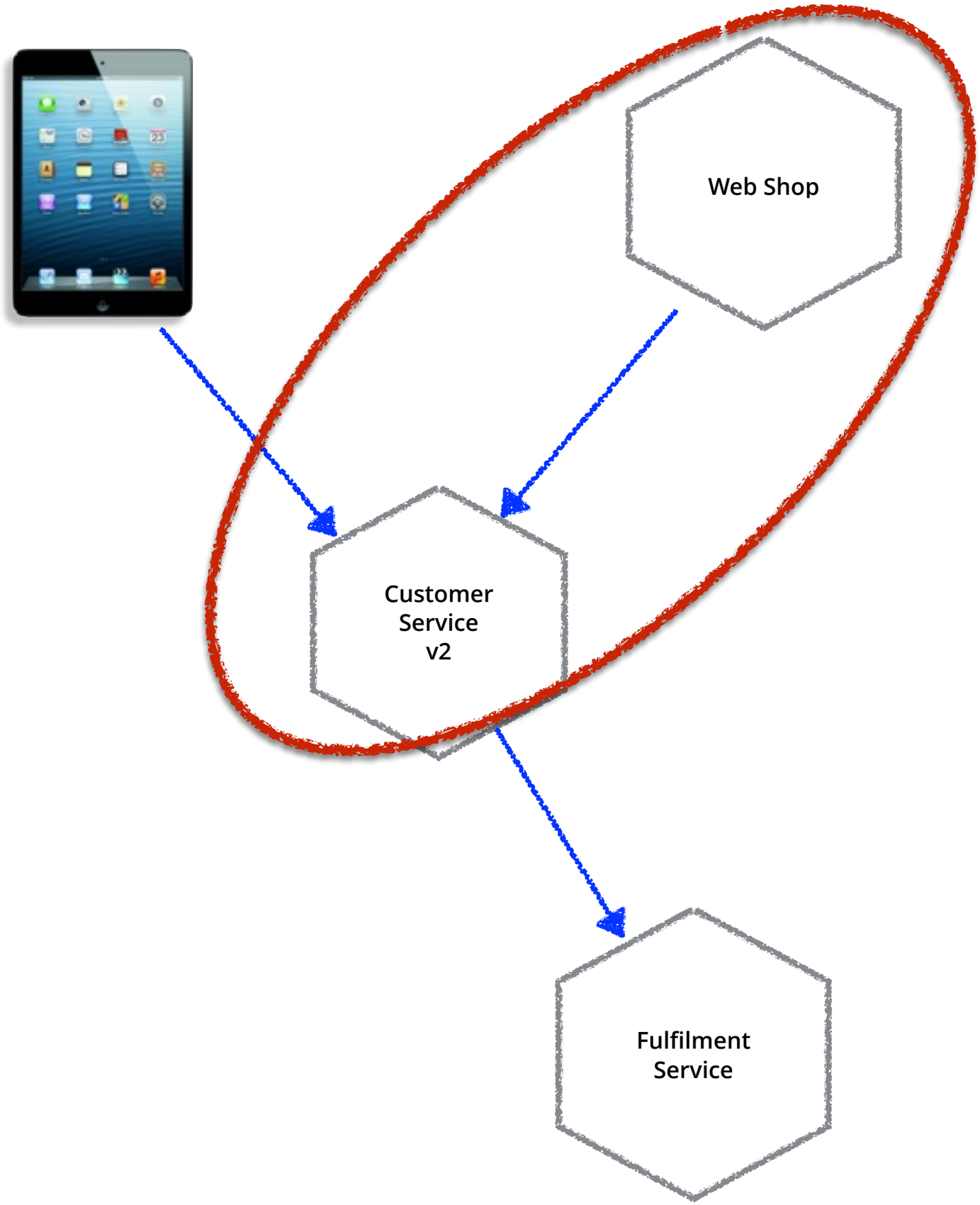

Build
S/M Tests
Large Tests

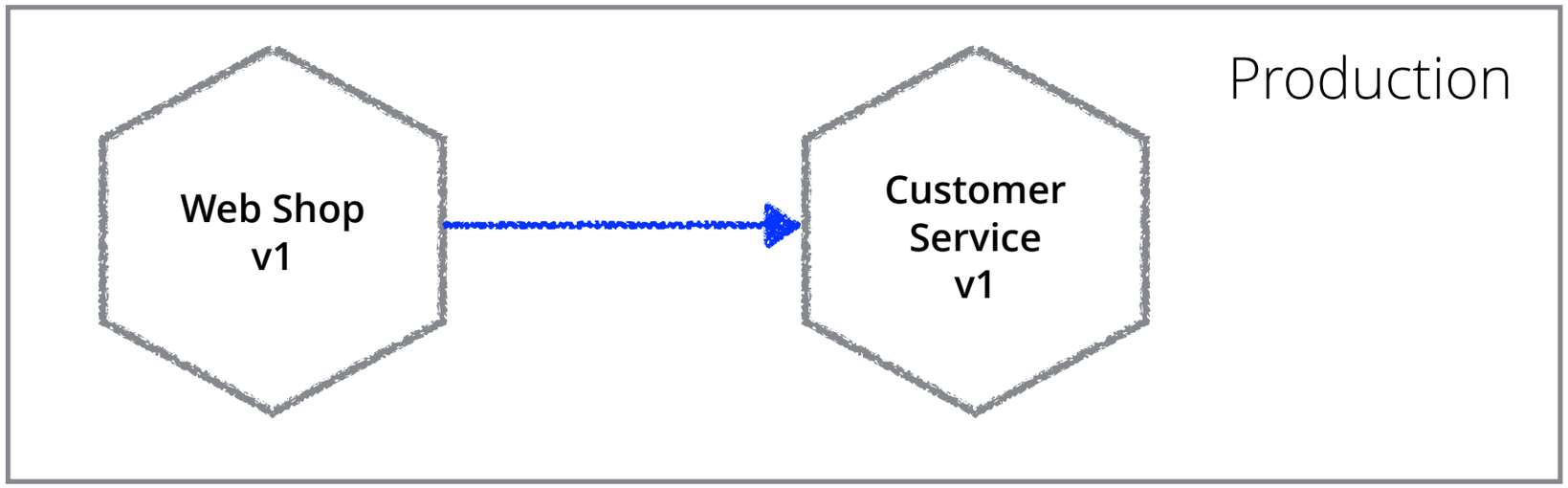

Build
S/M Tests
Large Tests
Customer Service
Fulfilment Service Stub


Web Shop
Customer
Service
V1
Fulfilment
Service

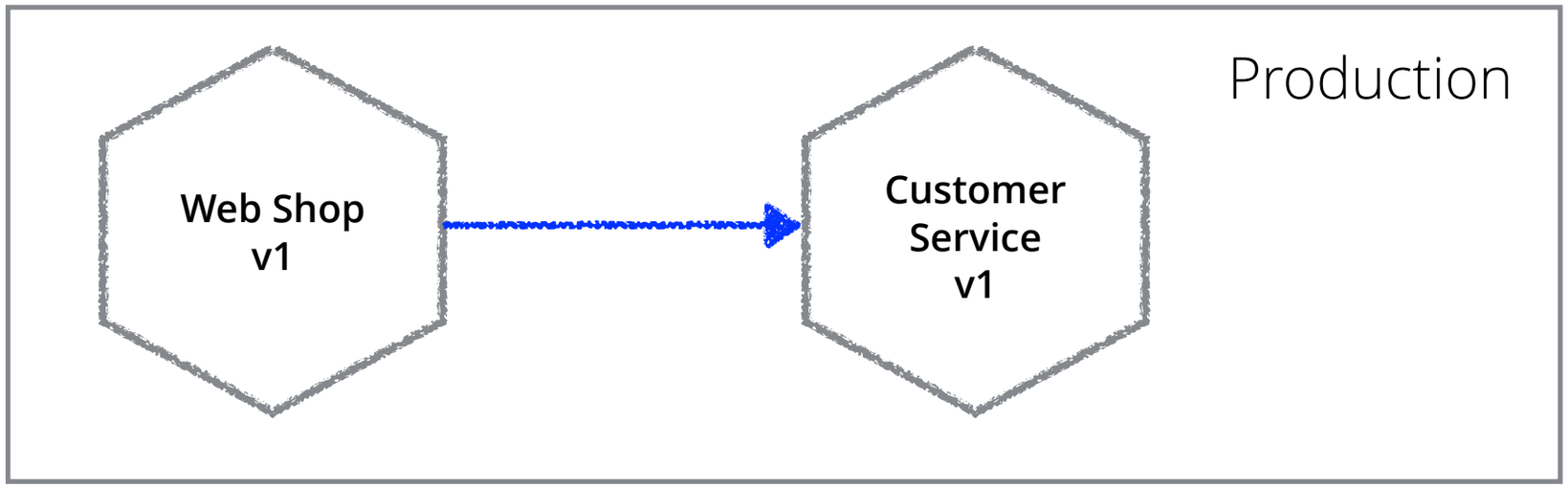

Web Shop
Customer
Service
v2
Fulfilment
Service


Web Shop
Customer
Service
v2
Fulfilment
Service

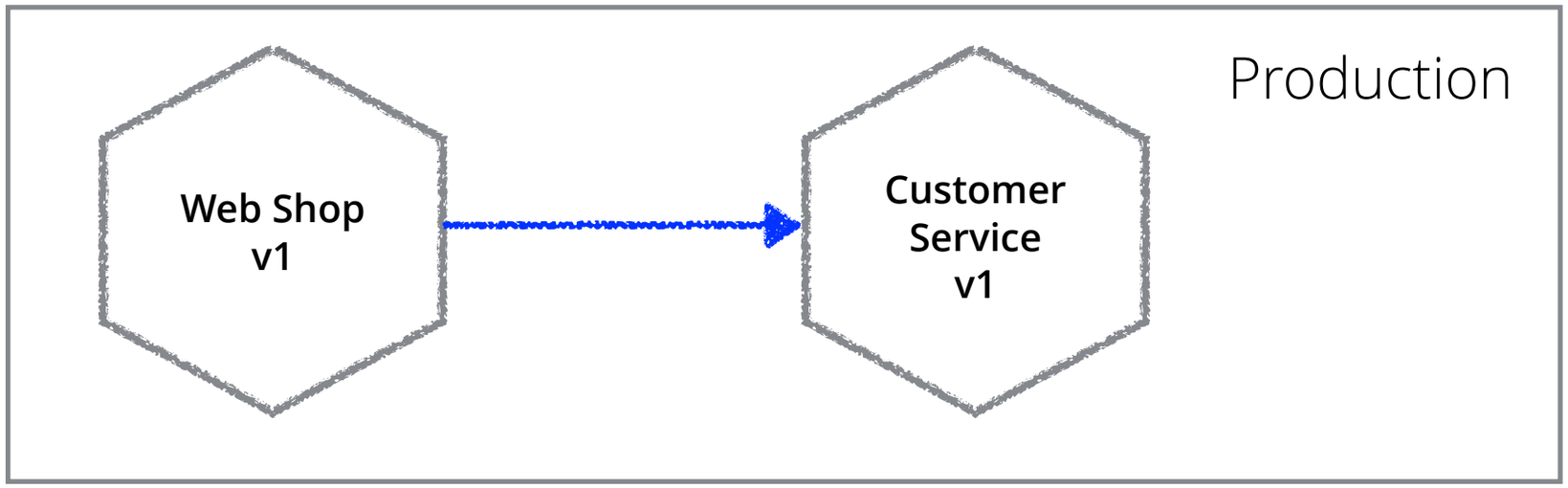

Web Shop
Customer Service v2
Fulfilment Service


Production
Web Shop
v1
Customer
Service
v1

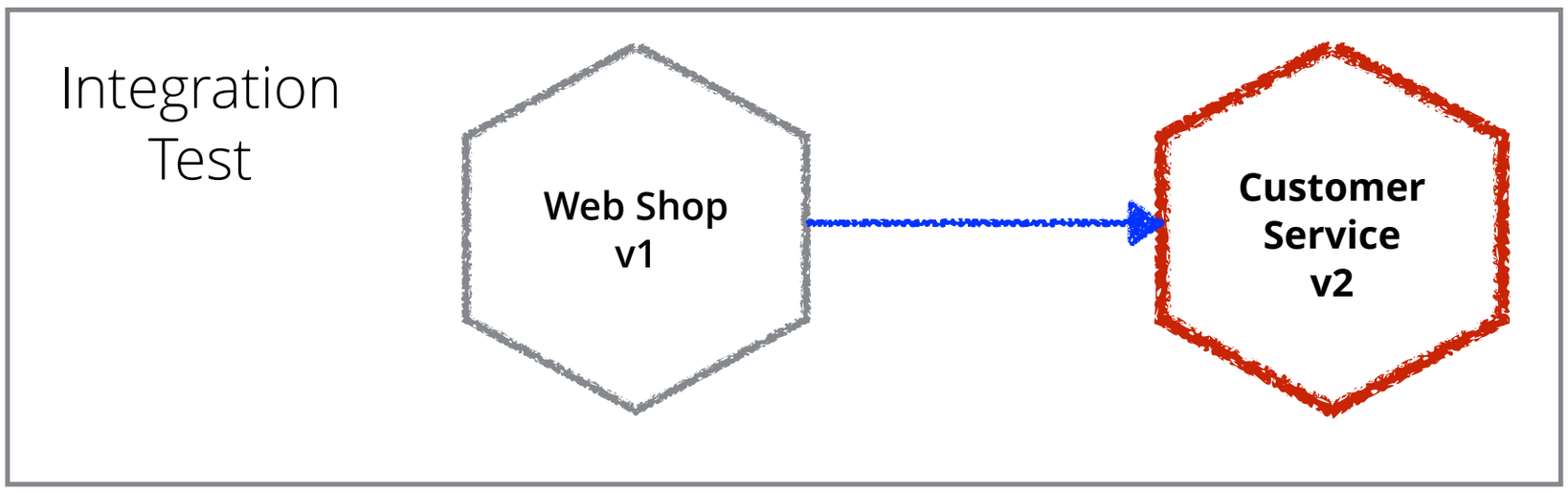

Customer
Service
Build
S/M Tests
Large Tests

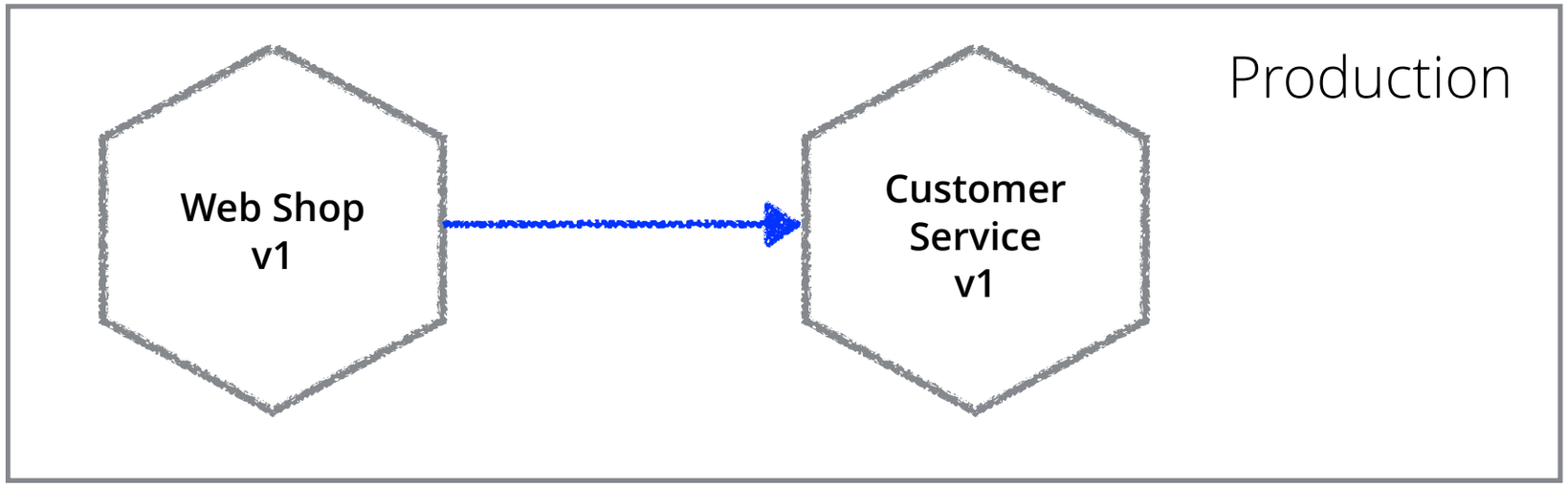

Production
Web Shop
v1
Customer
Service
v1

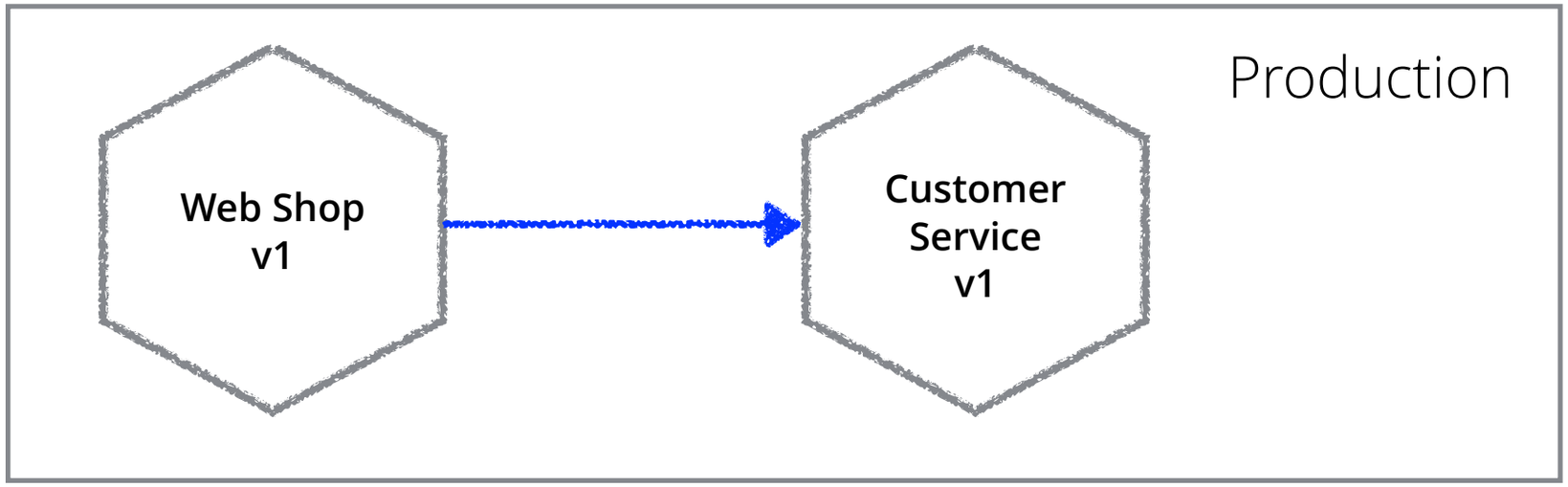

Customer
Service
Build
S/M Tests
Large Tests
Integration Test

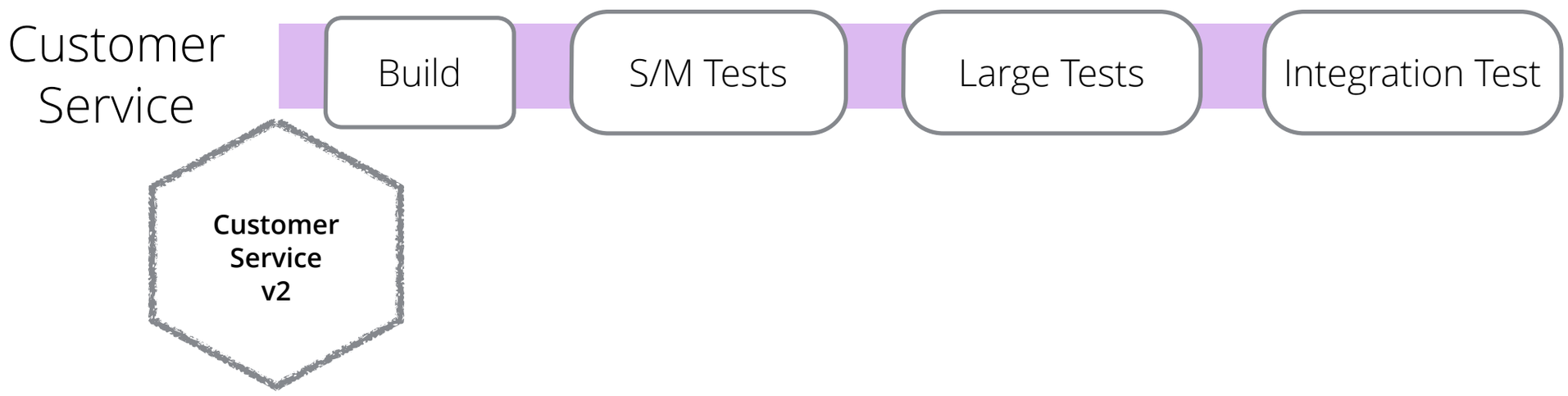

Production
Web Shop
v1
Customer
Service
v1

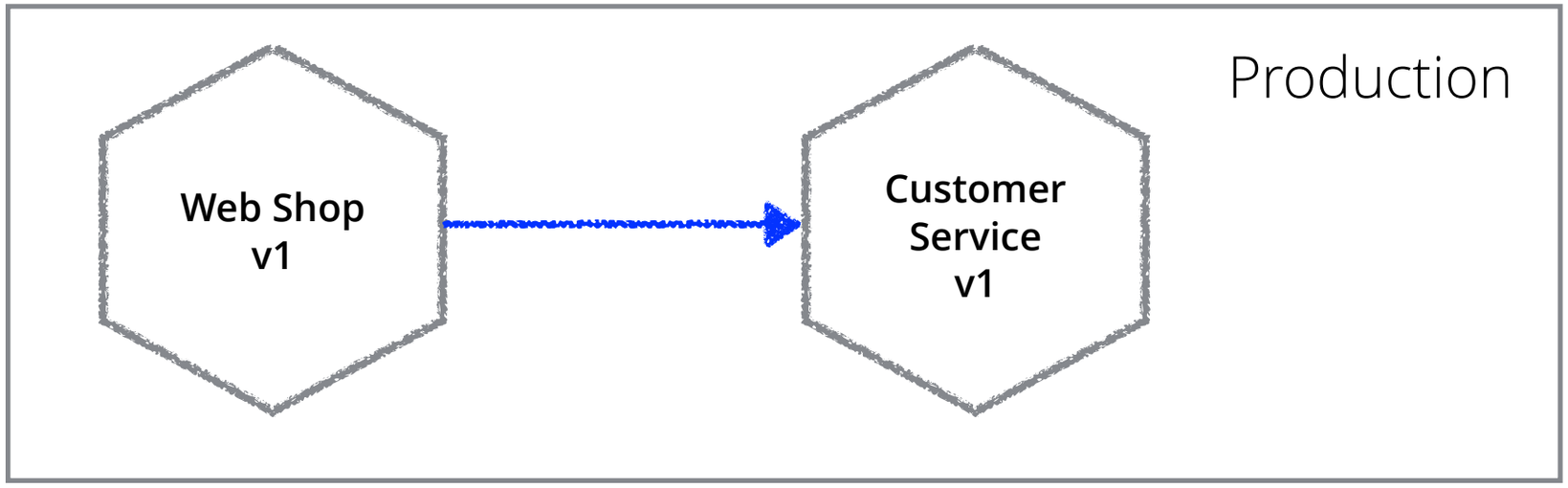

Customer
Service
Build
S/M Tests
Large Tests
Integration Test


Integration
Test
Web Shop
v1
Customer
Service
v2

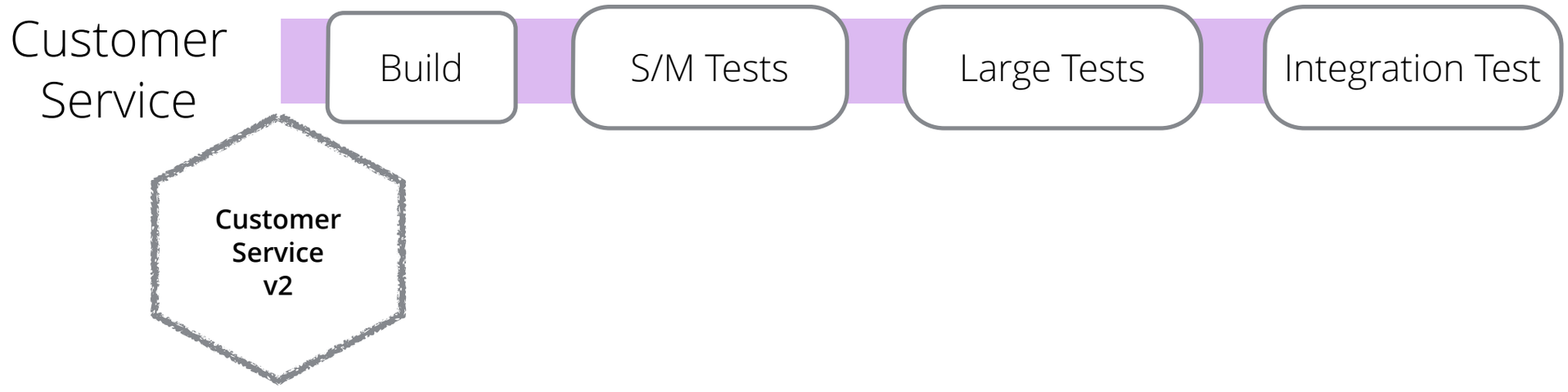

Production
Web Shop
v1
Customer
Service
v1

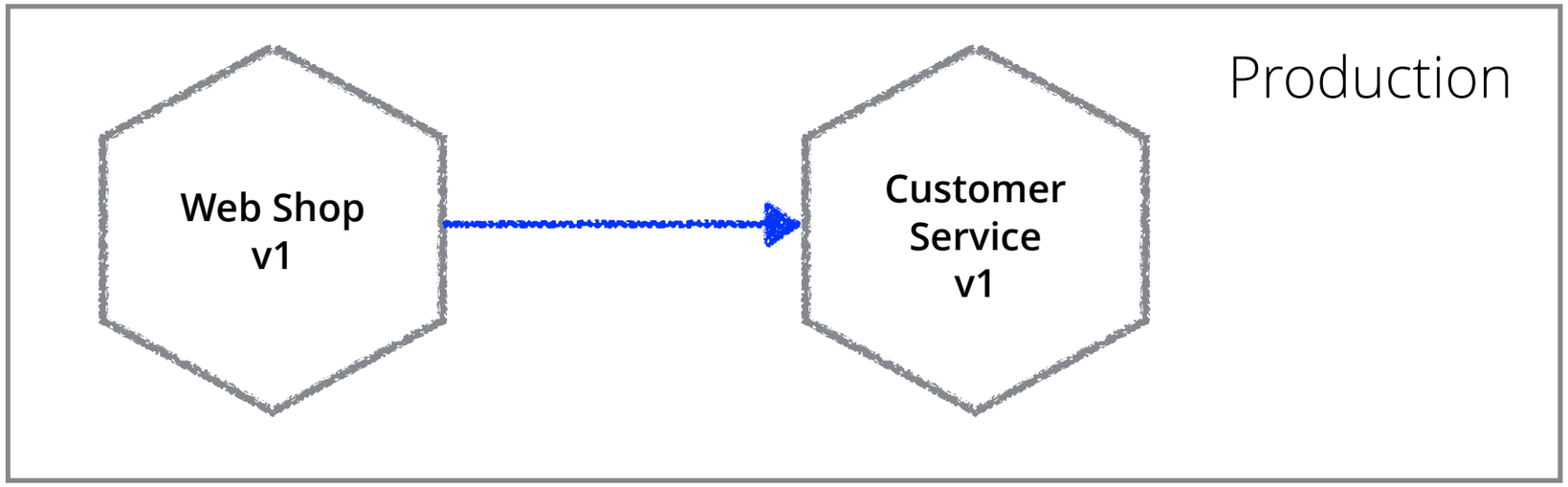

Production
Web Shop
v1
Customer
Service
v1

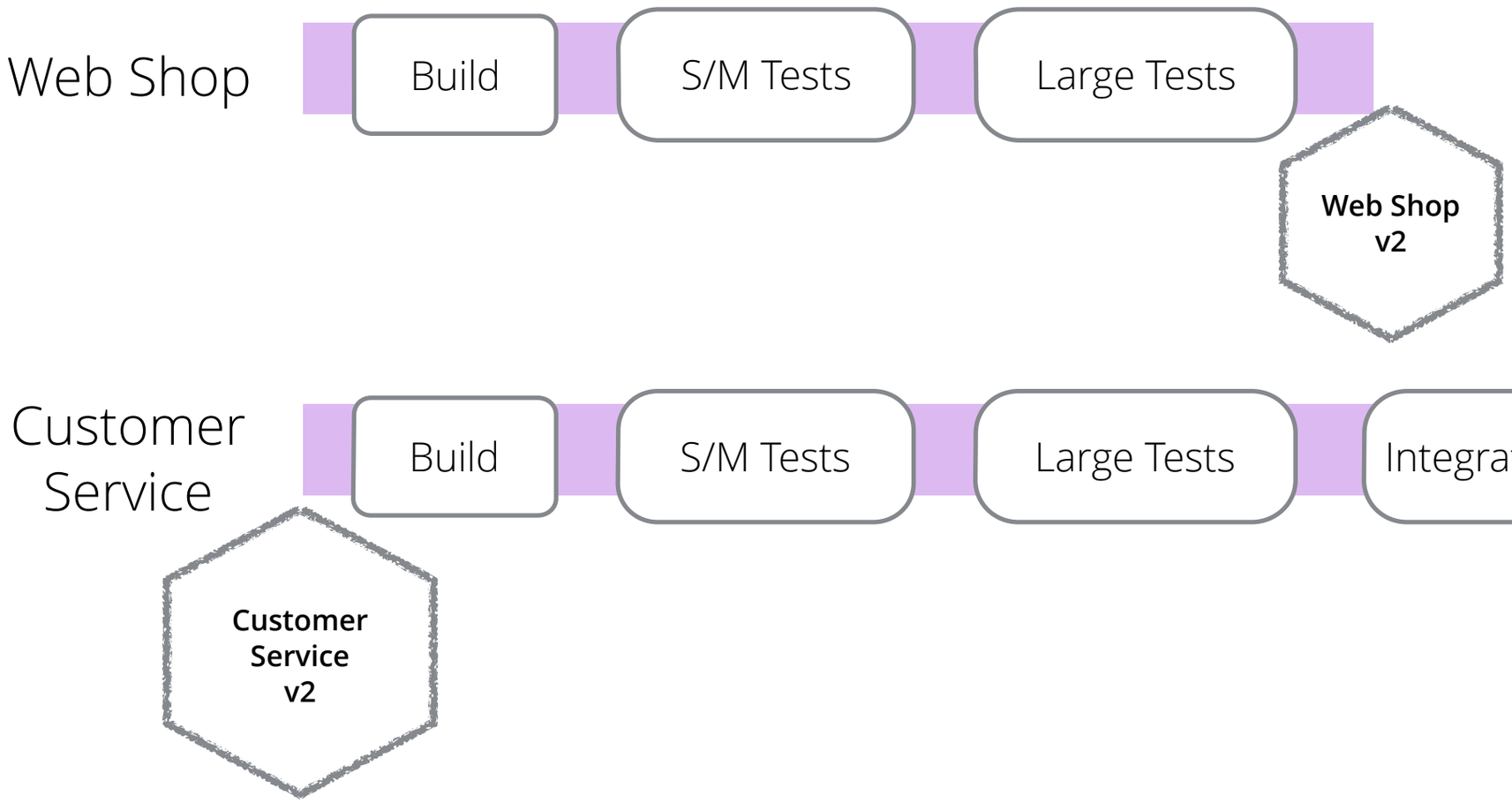

Customer
Service
Build
S/M Tests
Large Tests
Integration Test
Customer
Service
v2

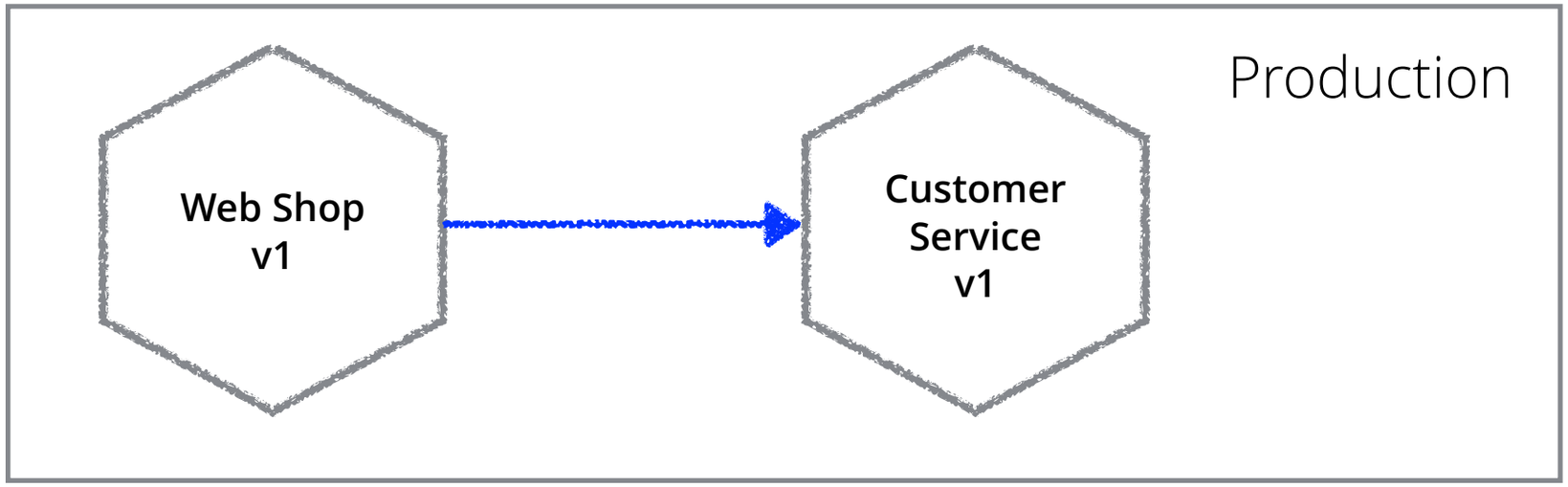

Production
Web Shop
v1
Customer
Service
v1

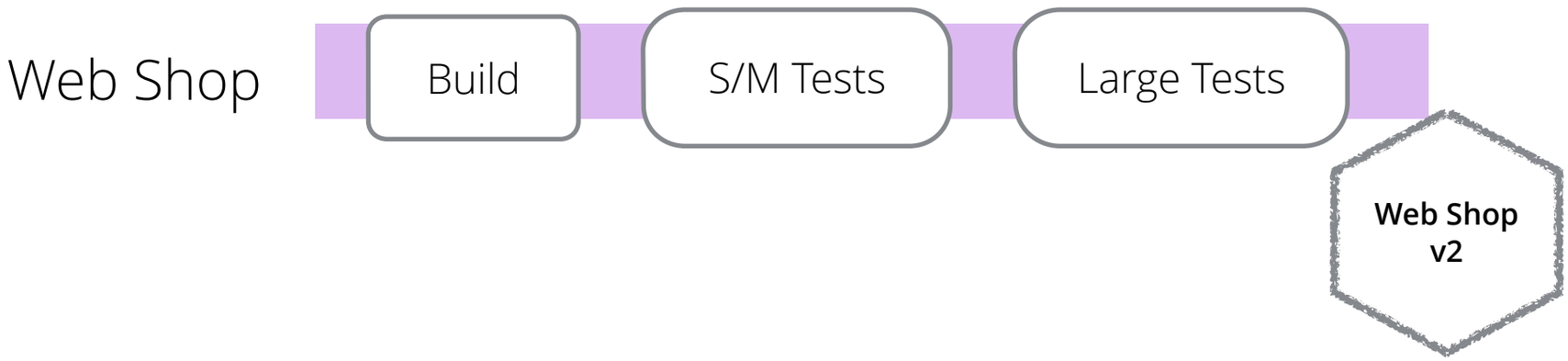

Web Shop
Build
S/M Tests
Large Tests

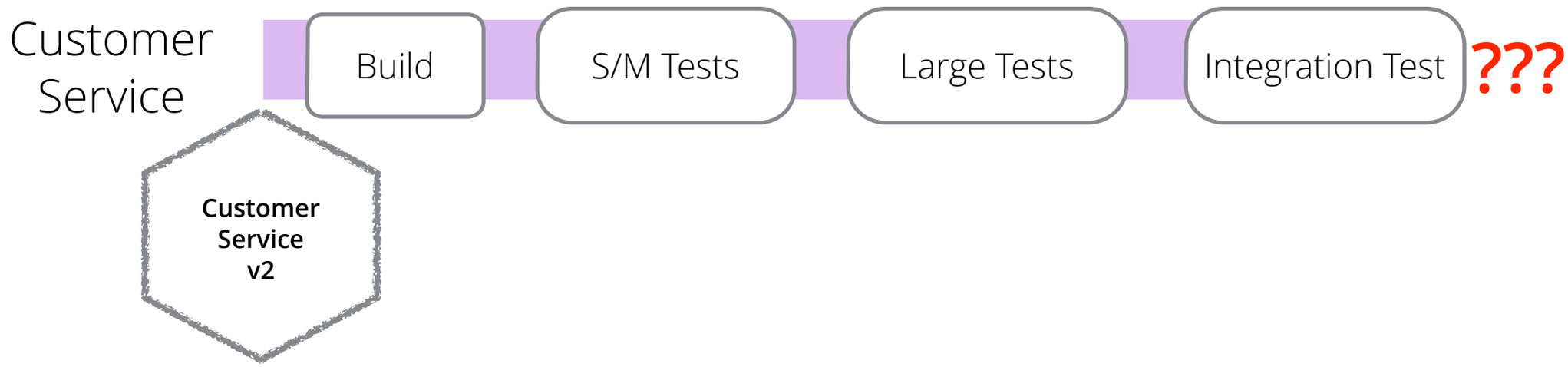

Customer
Service
Build
S/M Tests
Large Tests
Integration Test
Customer
Service
v2

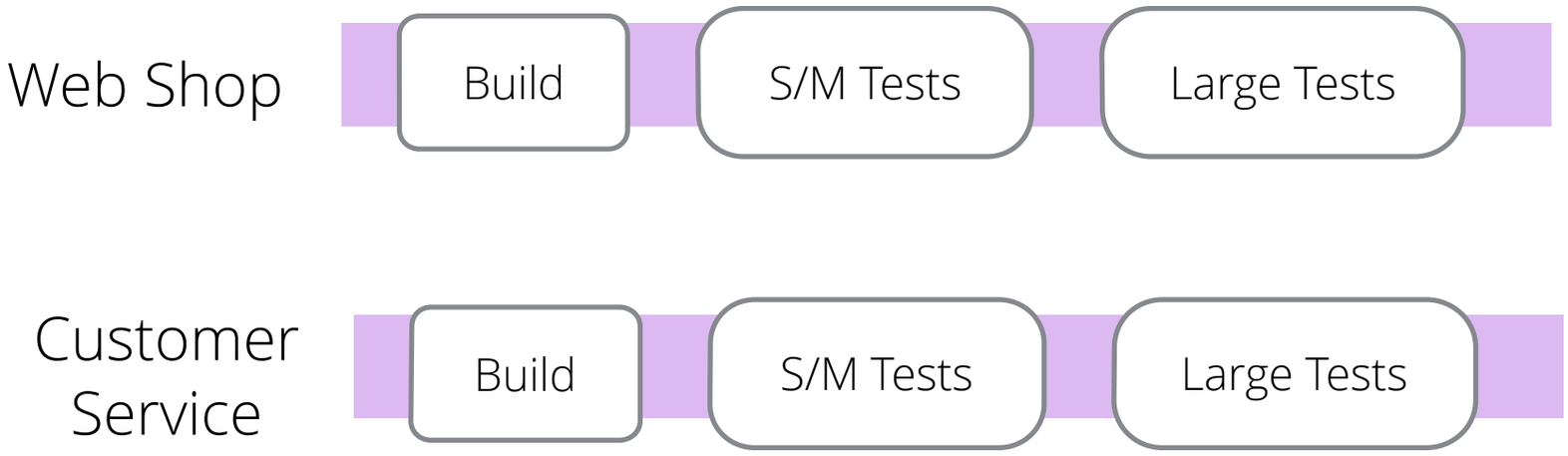

Production
Web Shop
v1
Customer
Service
v1

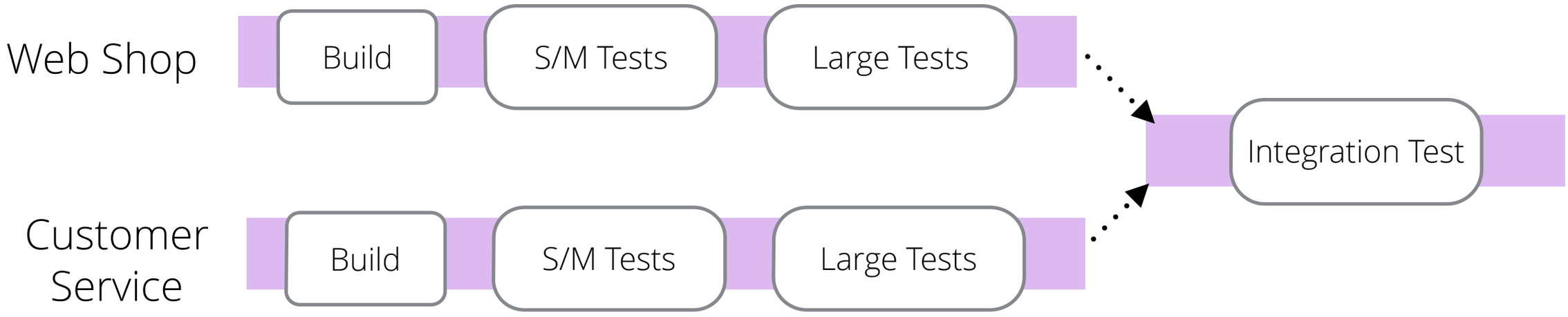

Web Shop
Build
S/M Tests
Large Tests
Web Shop
v2
Customer
Service
Build
S/M Tests
Large Tests
Integration Test
Customer
Service
v2

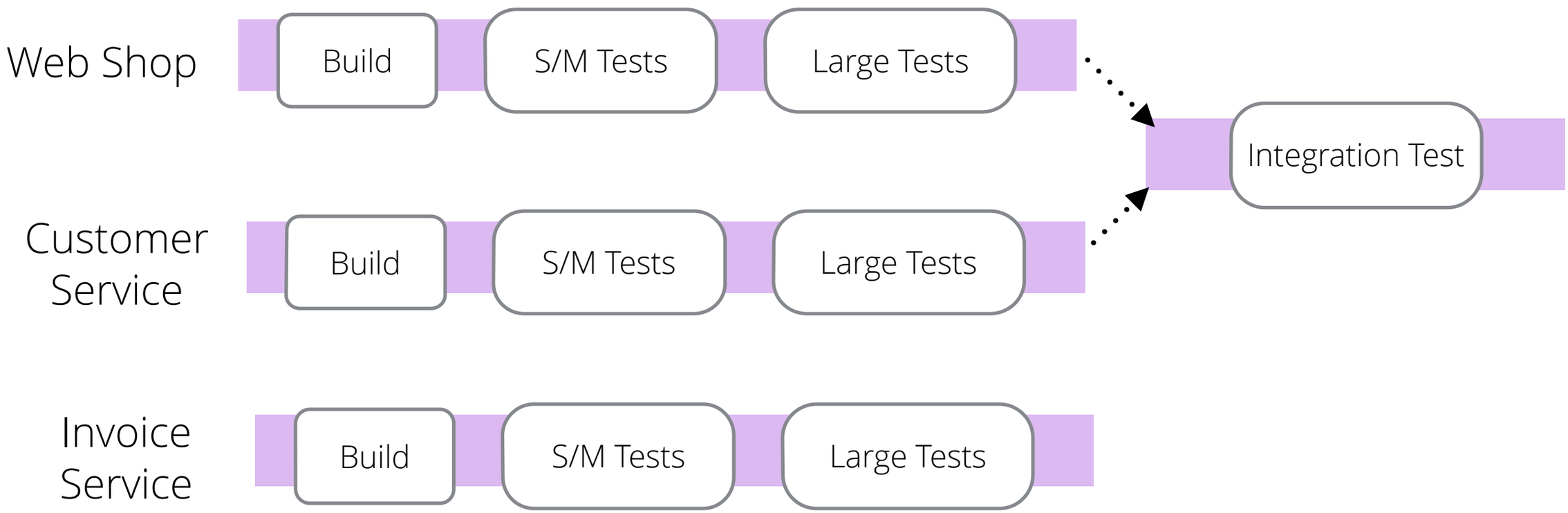

Production
Web Shop
v1
Customer
Service
v1

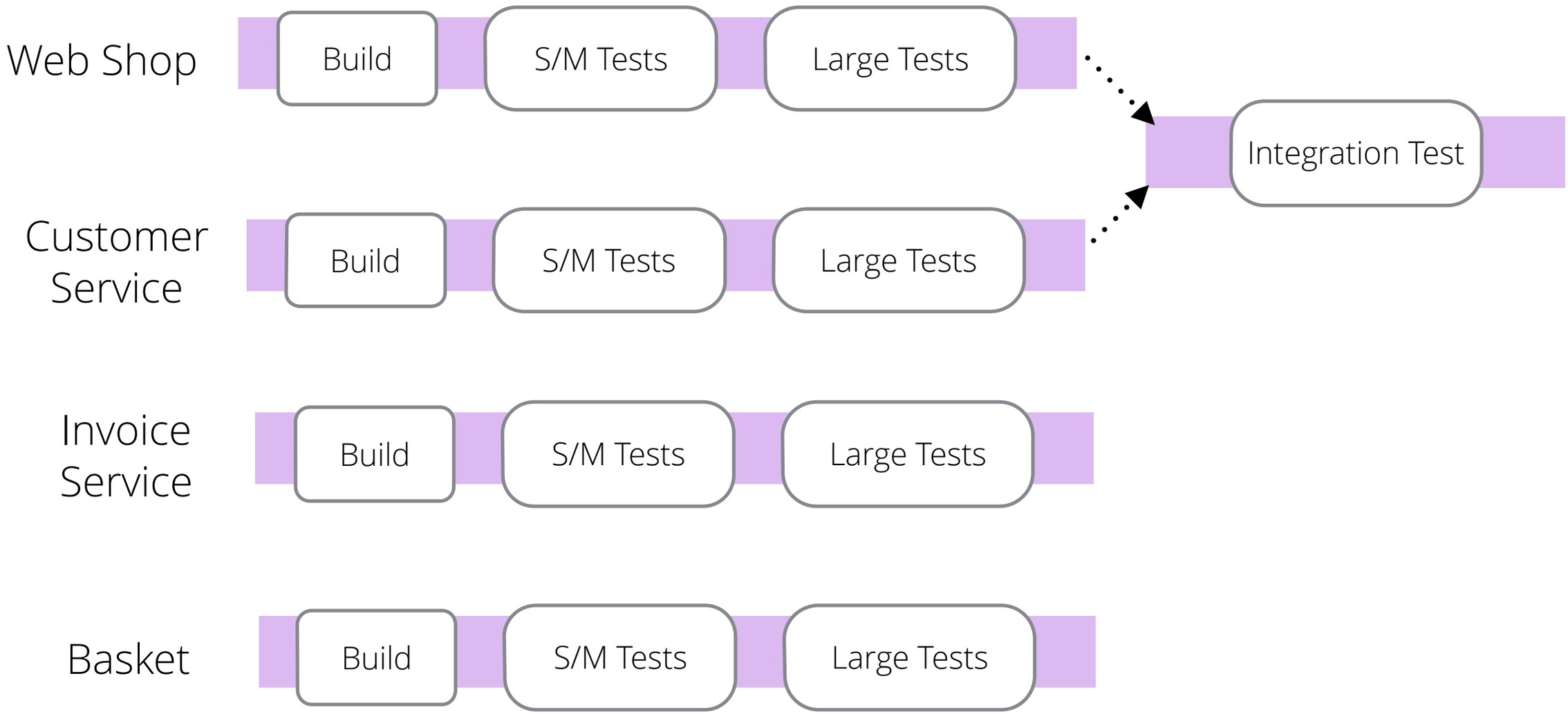

Web Shop
Build
S/M Tests
Large Tests
Web Shop
v2

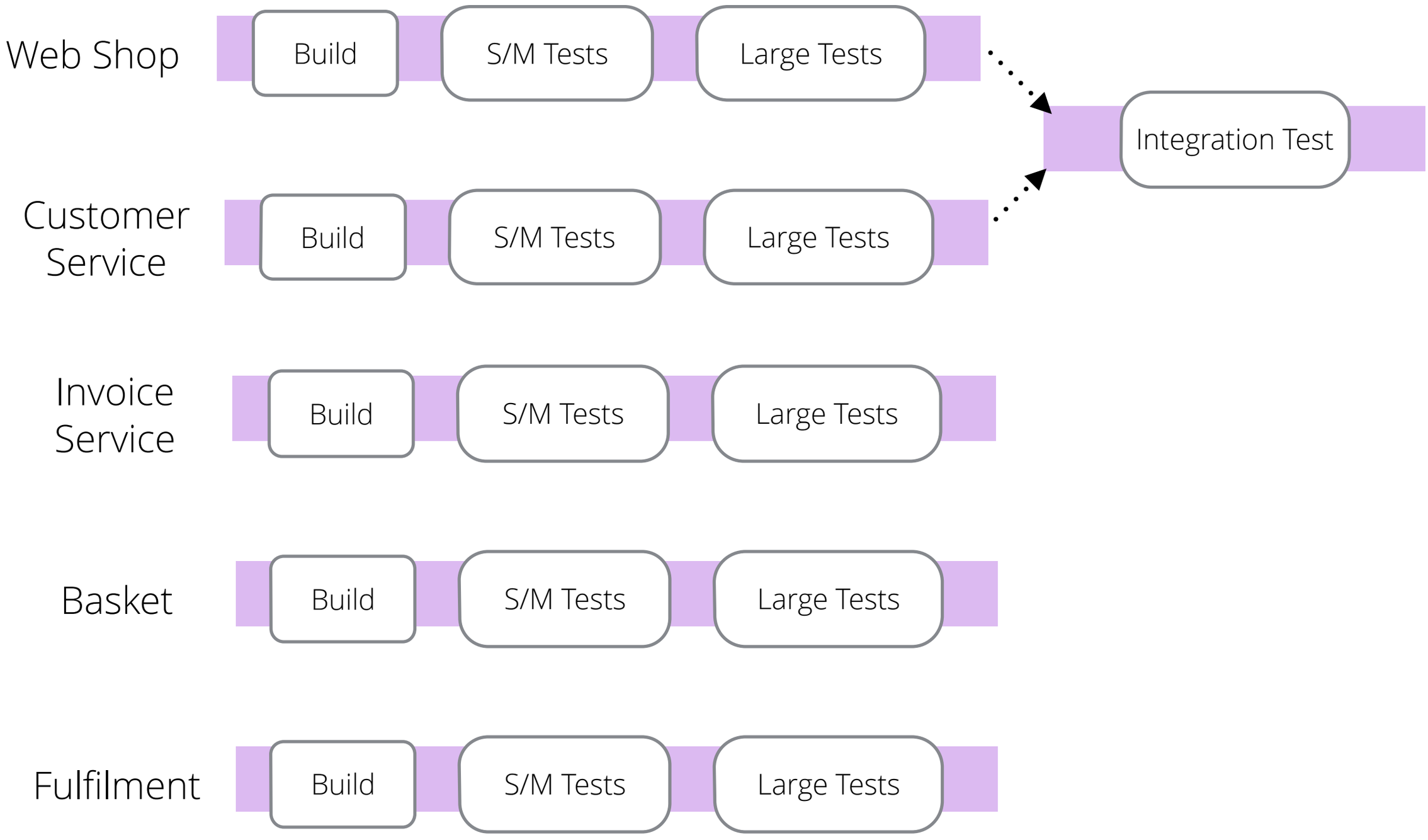

Customer
Service
Build
S/M Tests
Large Tests
Integration Test
???
Customer
Service
v2


Web Shop
Build
S/M Tests
Large Tests
Customer Service
Build
S/M Tests
Large Tests


Web Shop
Build
S/M Tests
Large Tests
Integration Test
Customer Service
Build
S/M Tests
Large Tests


Web Shop
Build
S/M Tests
Large Tests
Integration Test
Customer Service
Build
S/M Tests
Large Tests
Invoice Service
Build
S/M Tests
Large Tests


Web Shop
Build
S/M Tests
Large Tests
Integration Test
Customer Service
Build
S/M Tests
Large Tests
Invoice Service
Build
S/M Tests
Large Tests
Basket
Build
S/M Tests
Large Tests


Web Shop
Build
S/M Tests
Large Tests
Integration Test
Customer Service
Build
S/M Tests
Large Tests
Invoice Service
Build
S/M Tests
Large Tests
Basket
Build
S/M Tests
Large Tests
Fulfilment
Build
S/M Tests
Large Tests

Browsers

Timing

Browsers

Provisioning of Environments

Timing

Browsers

Networks

Provisioning of Environments

Timing

Browsers

Deployment

Networks

Provisioning of Environments

Timing

Browsers

Deployment

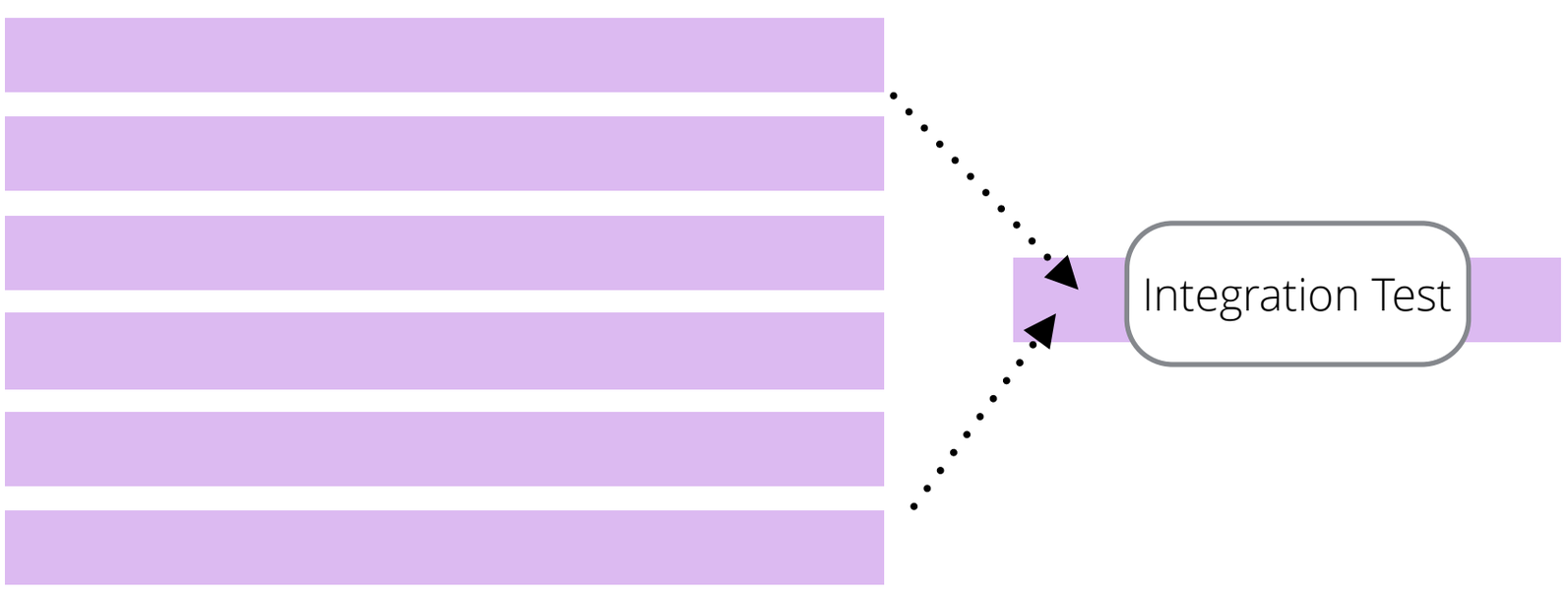
Networks

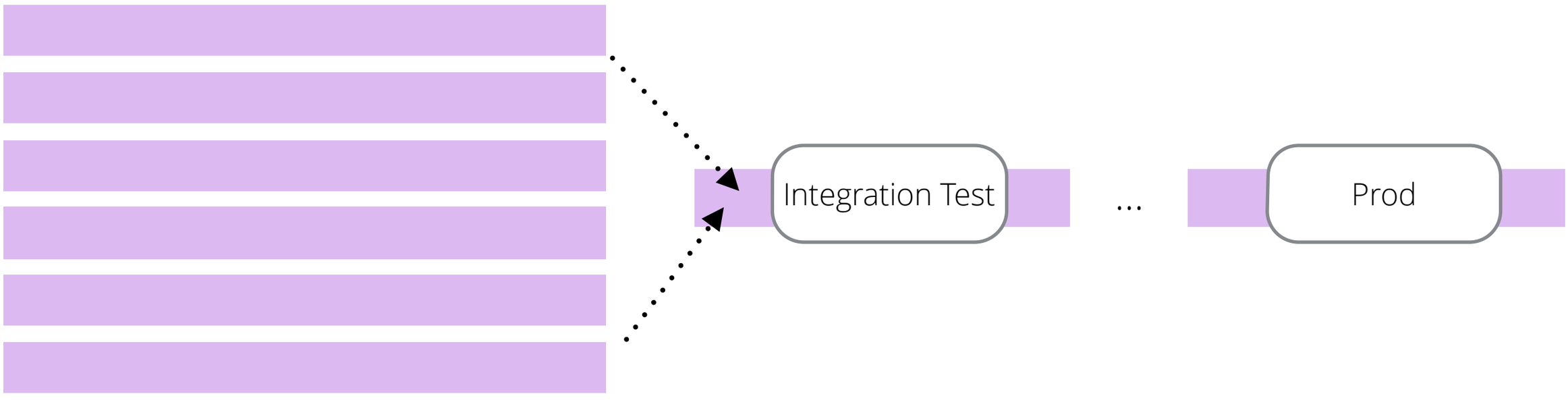
Provisioning of Environments

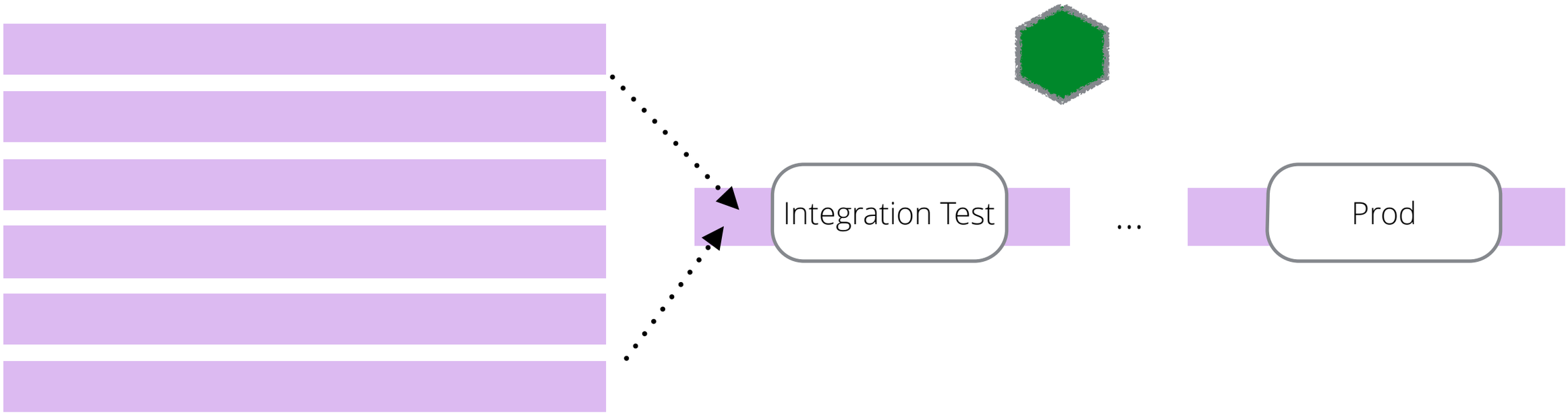
Timing

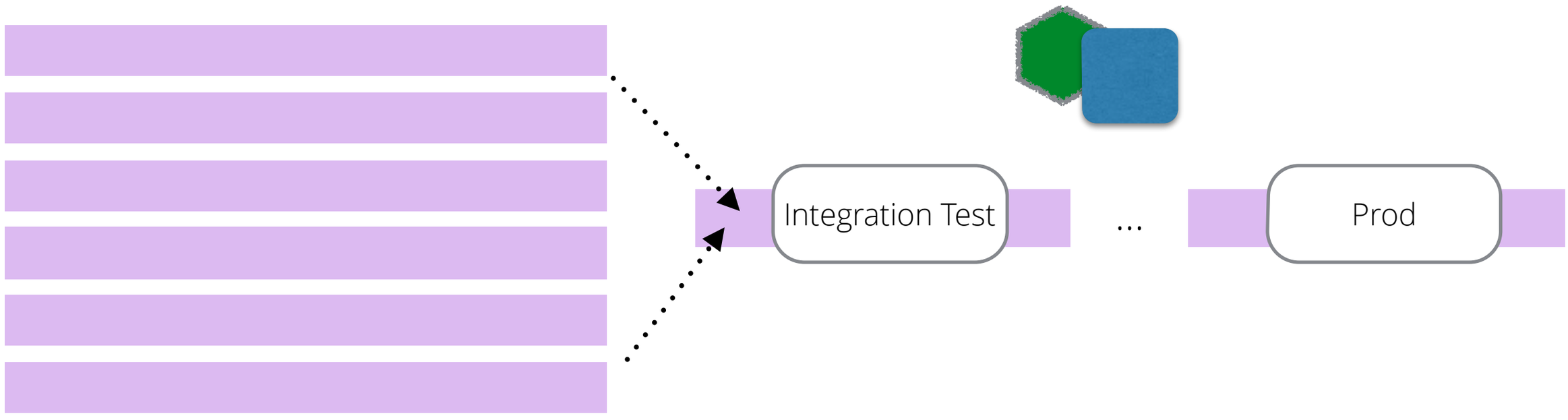
Browsers

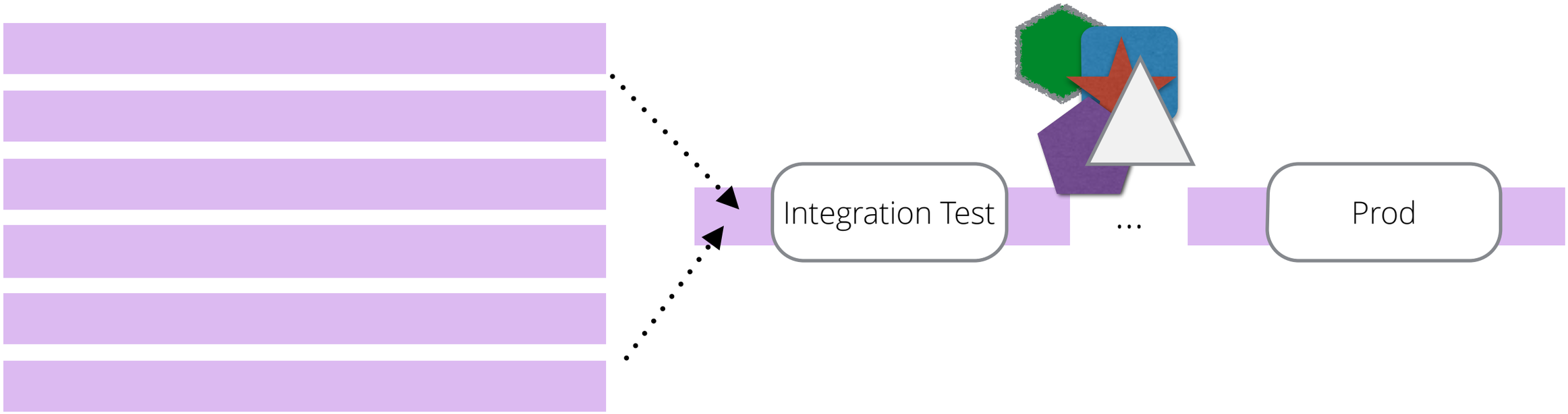
Diagnosis

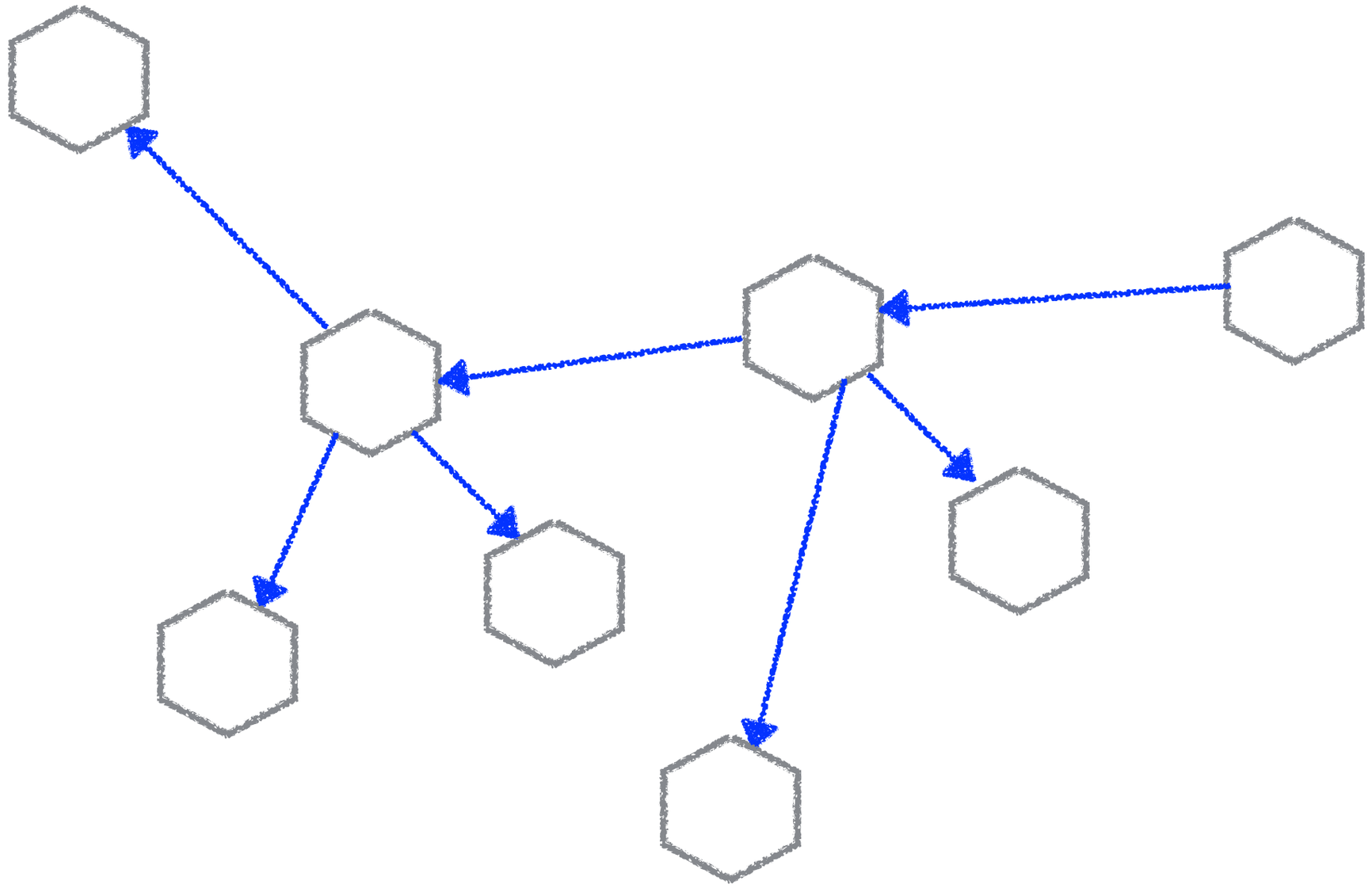

Integration Test

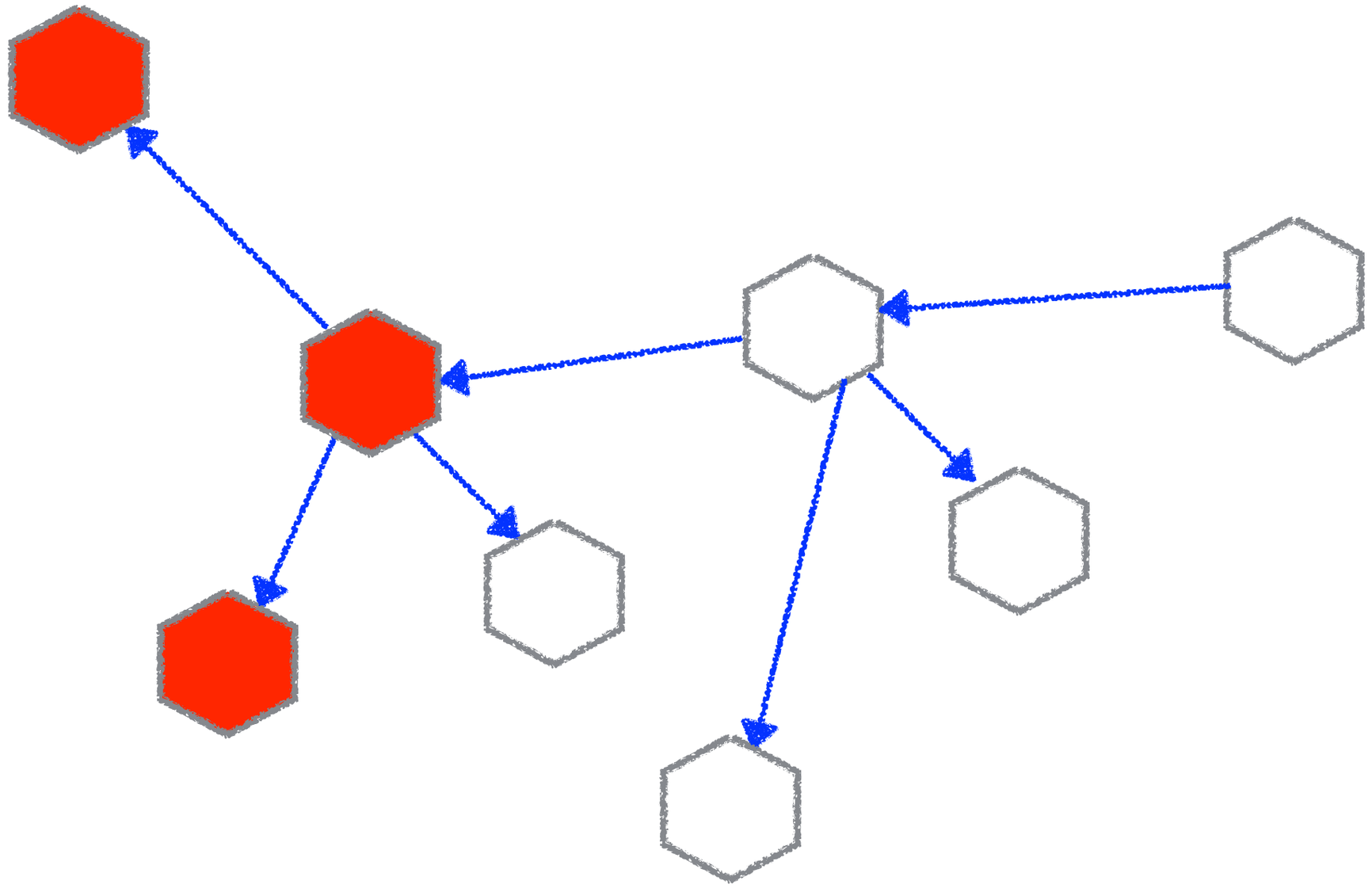

Integration Test
...
Prod

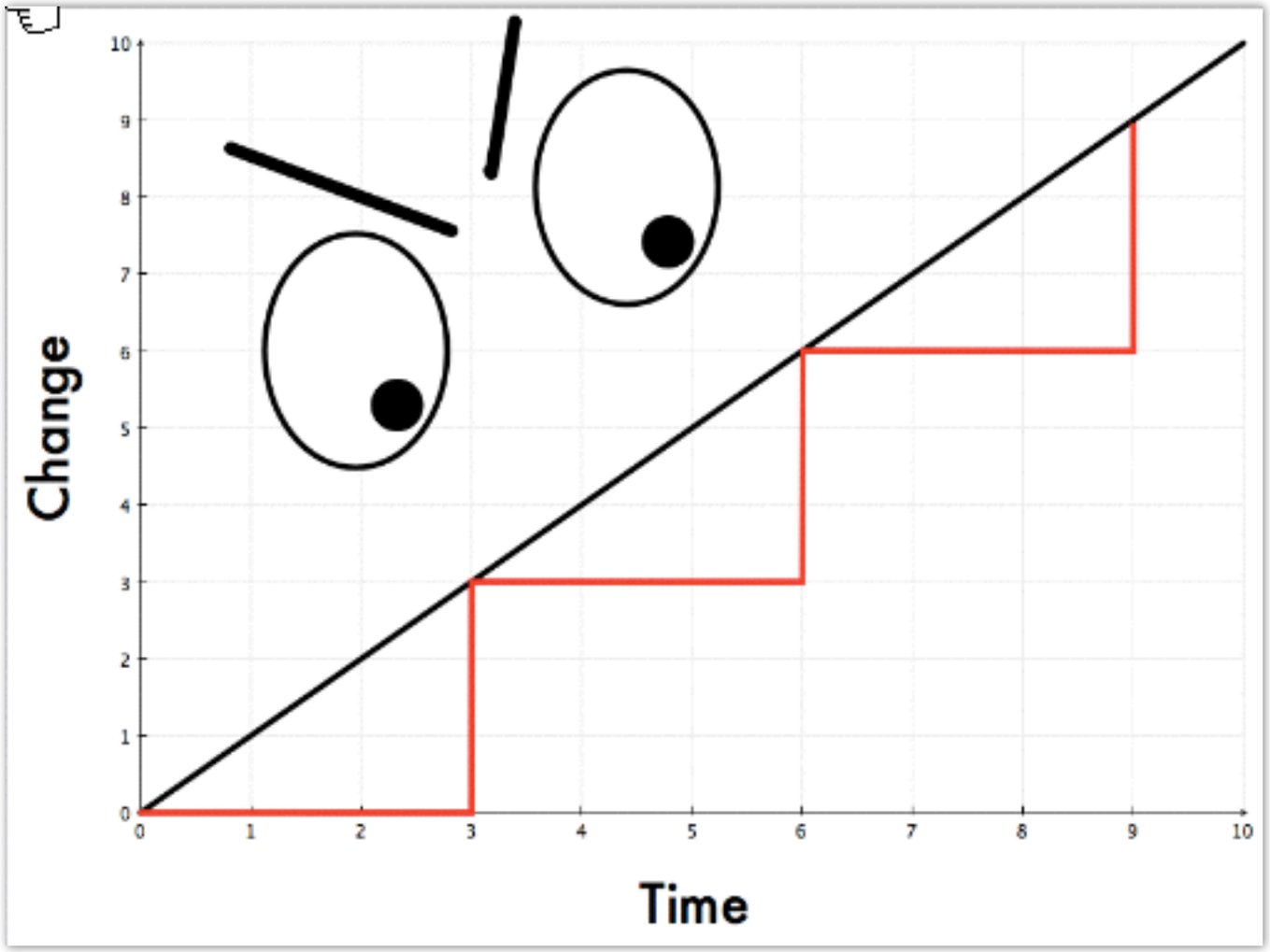

Integration Test
...
Prod


Integration Test
...
Prod

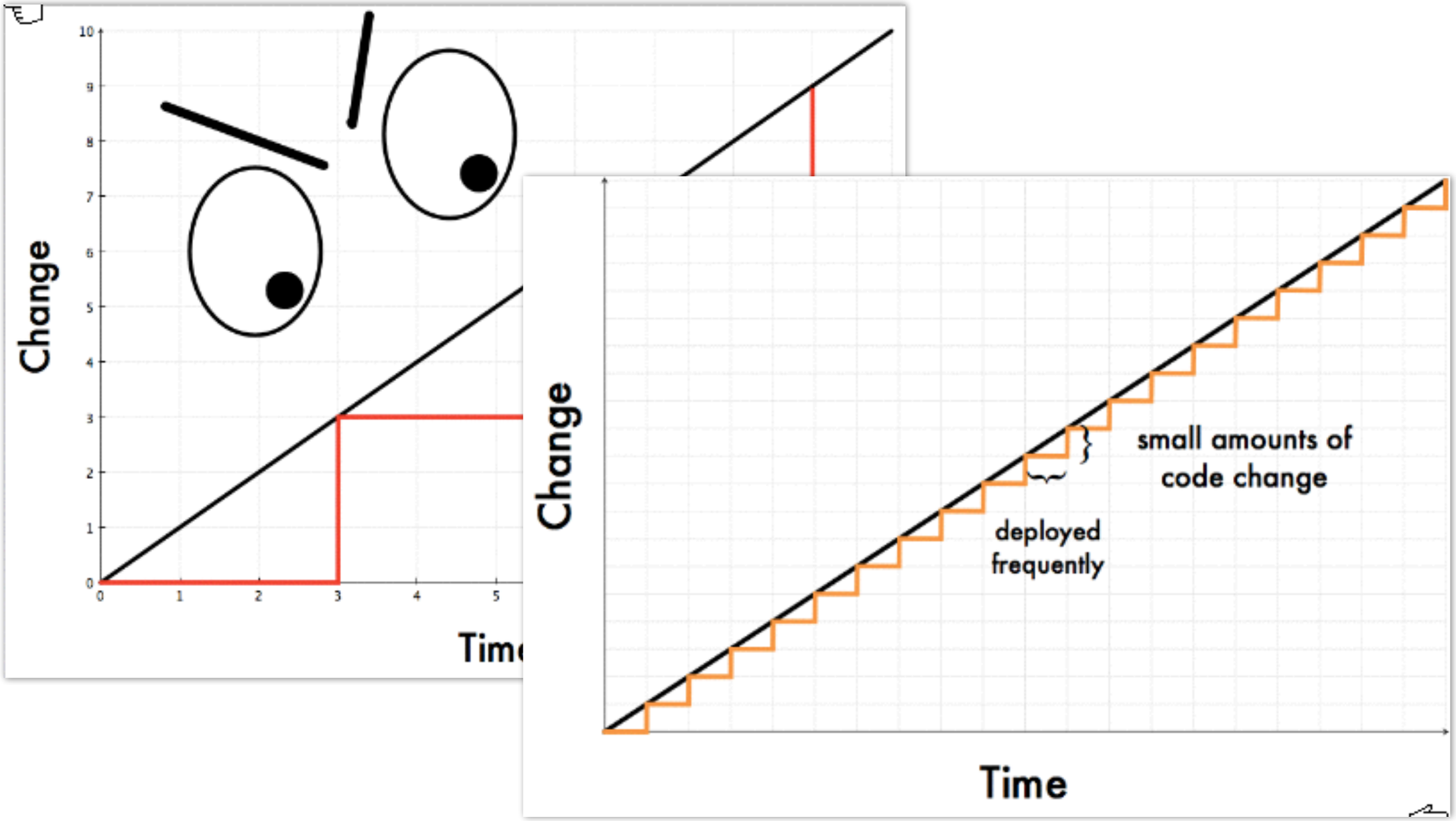

Integration Test
...
Prod



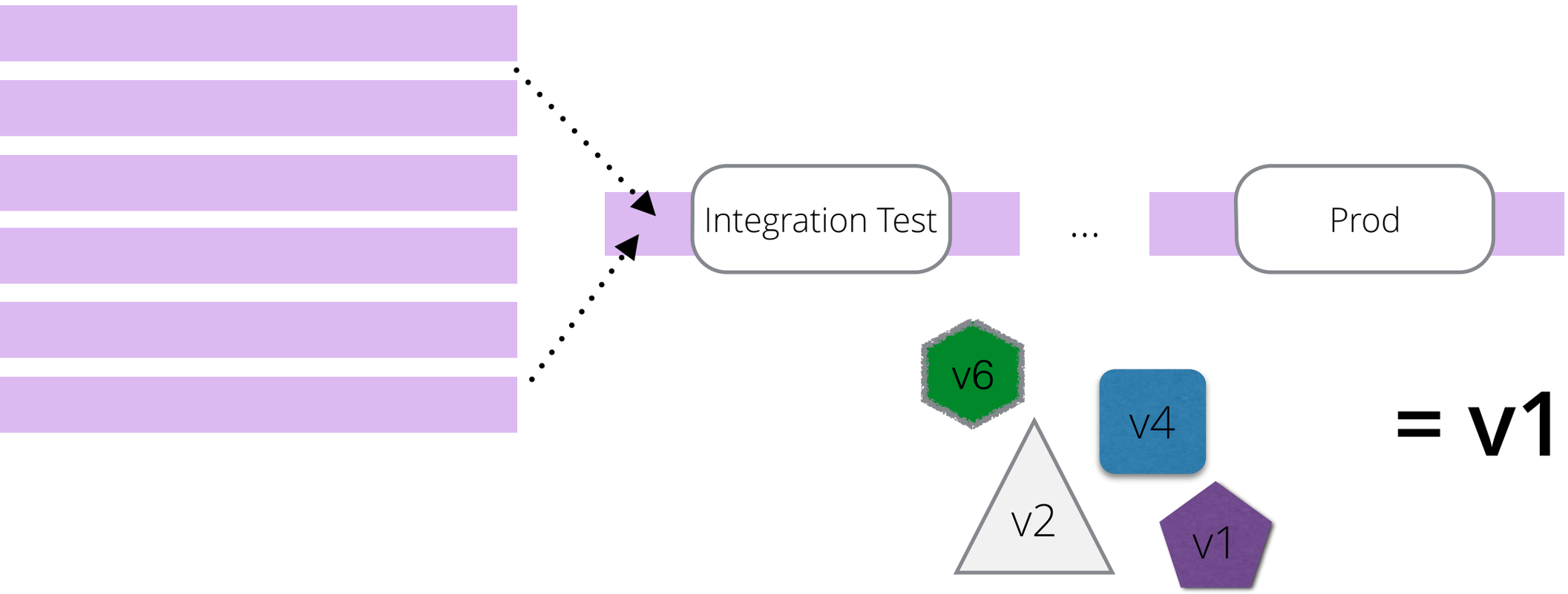




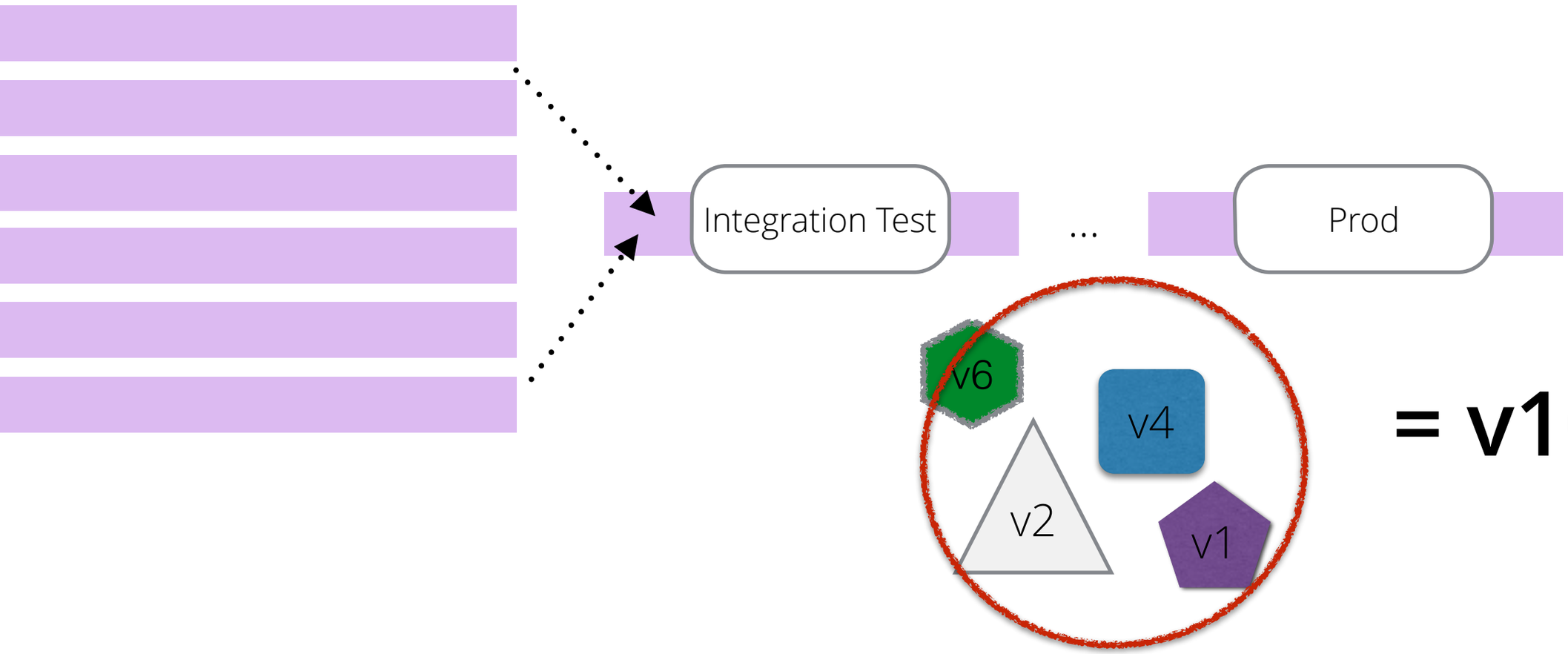
John Allspaw: “Ops Metametrics” http://slidesha.re/dsSZIr



John Allspaw: “Ops Metametrics” http://slidesha.re/dsSZIr

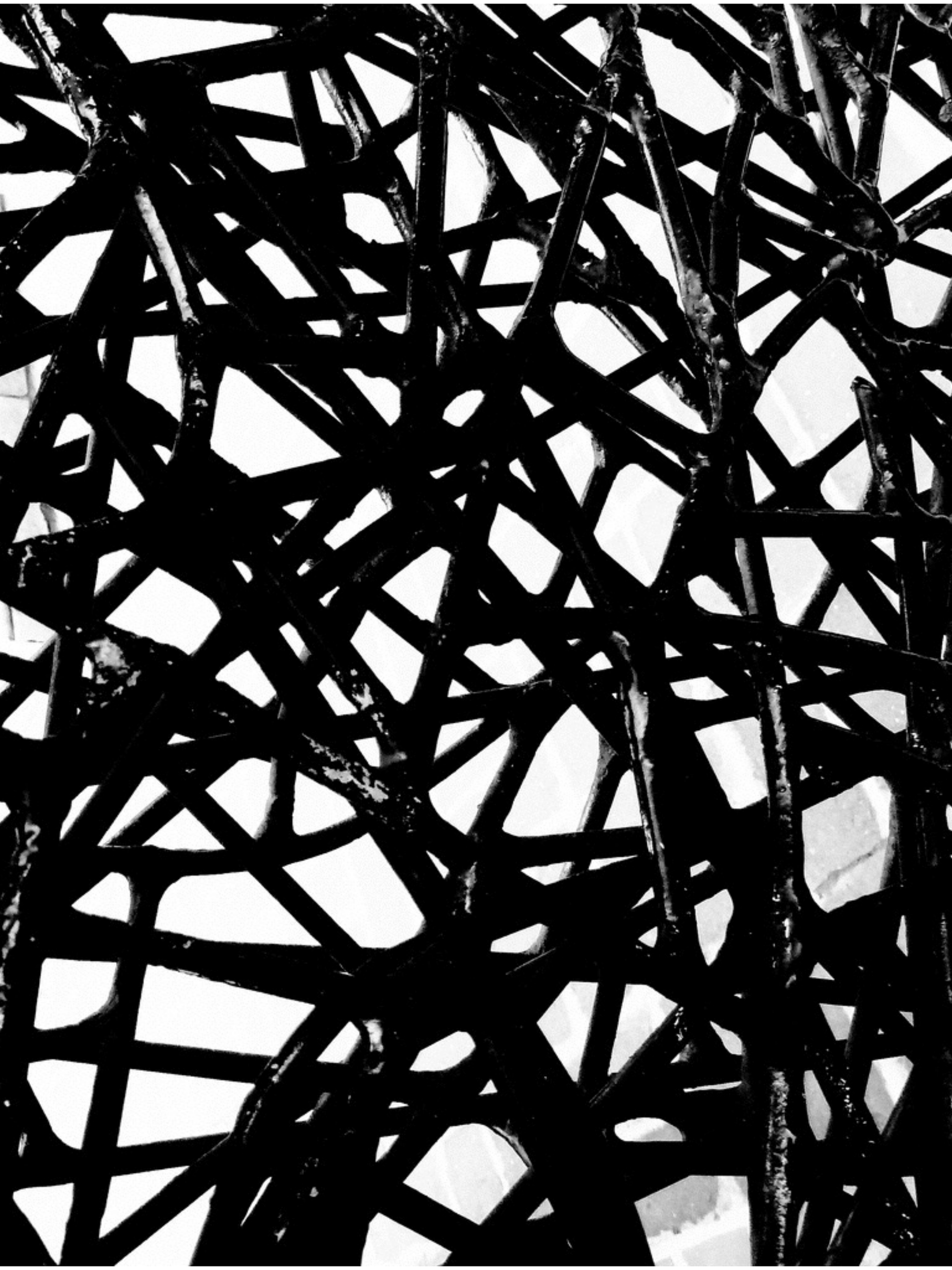


= v10



= v10

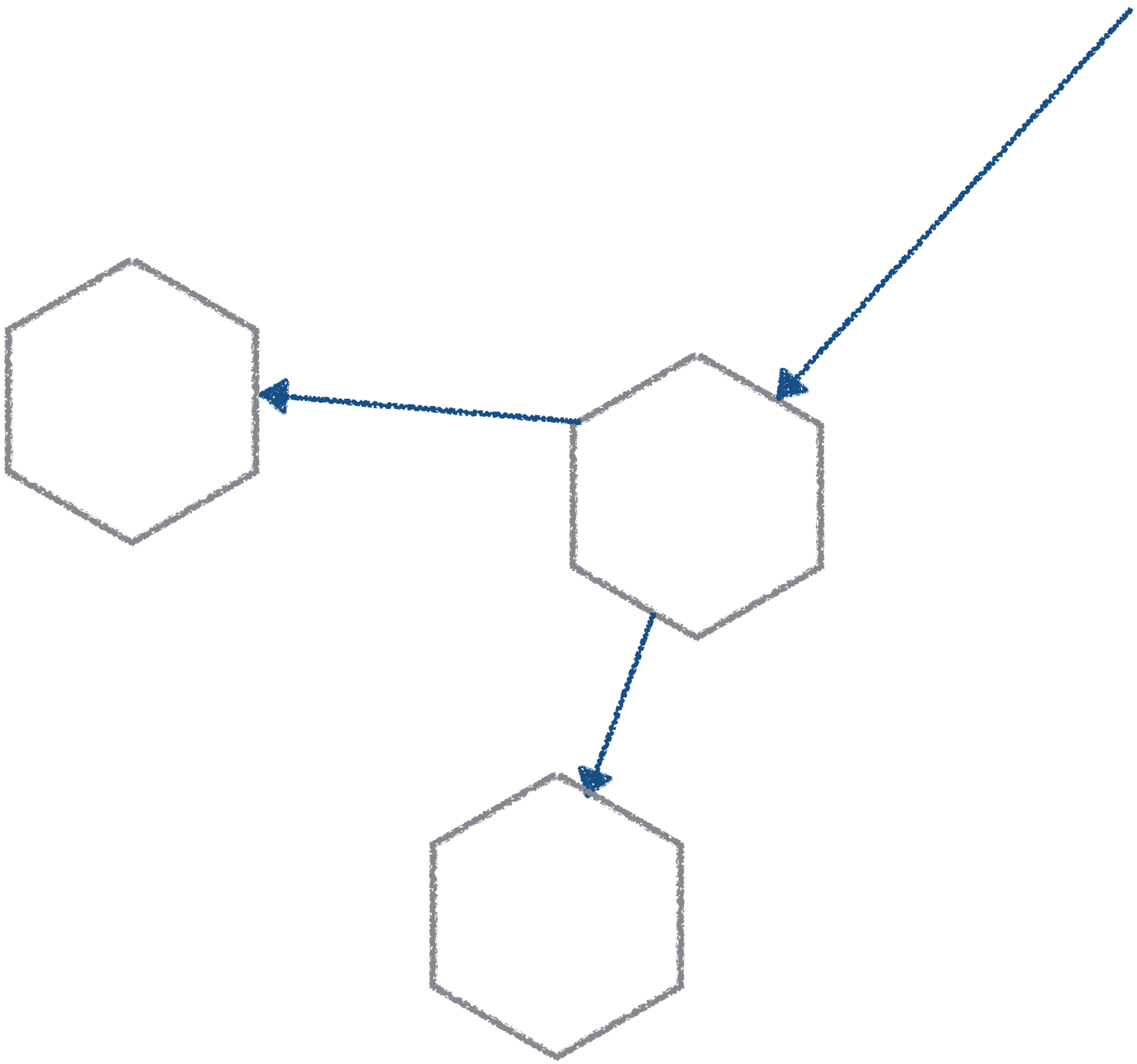
Danger Will Robinson!

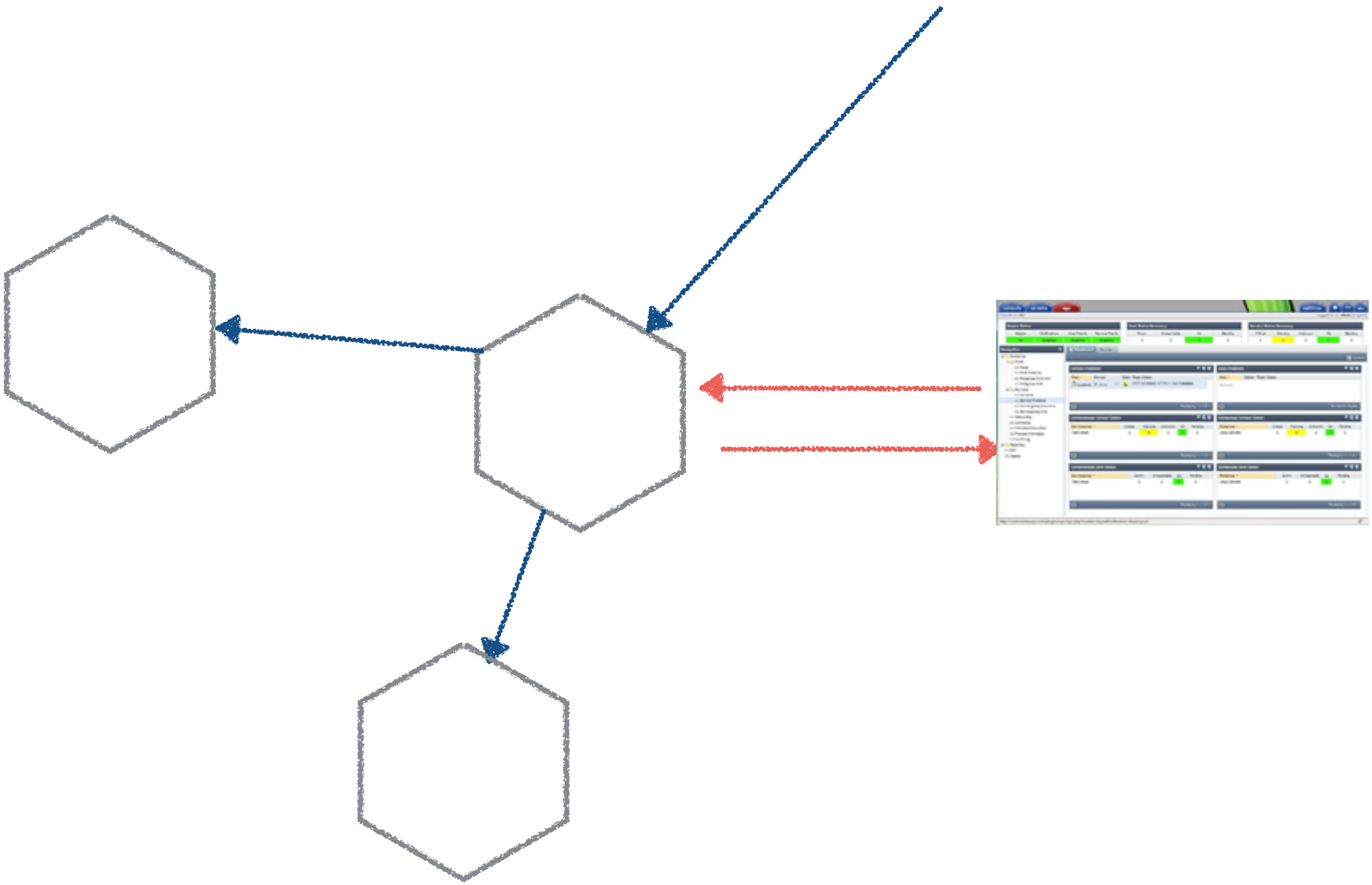


Golden Rule:

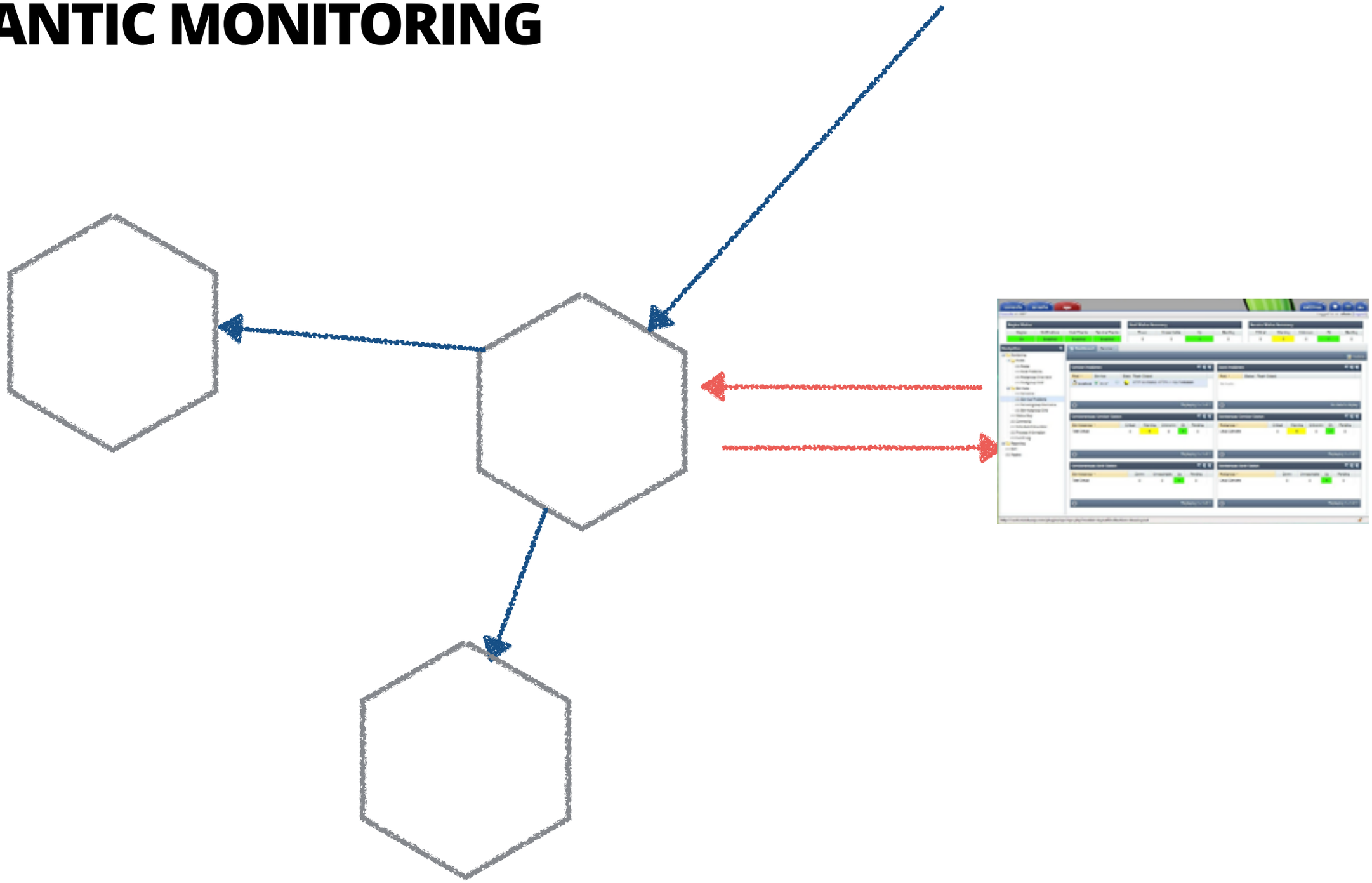
Get good at releasing services independently

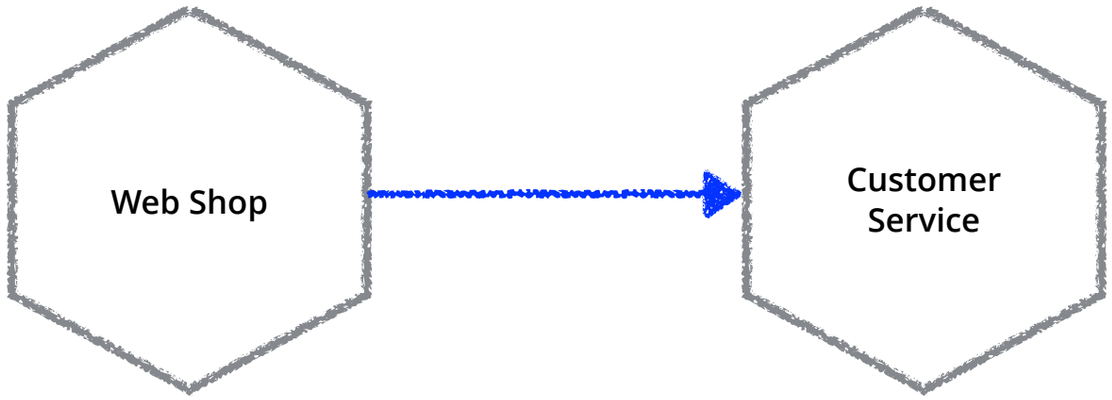
SO NO INTEGRATION TESTS?

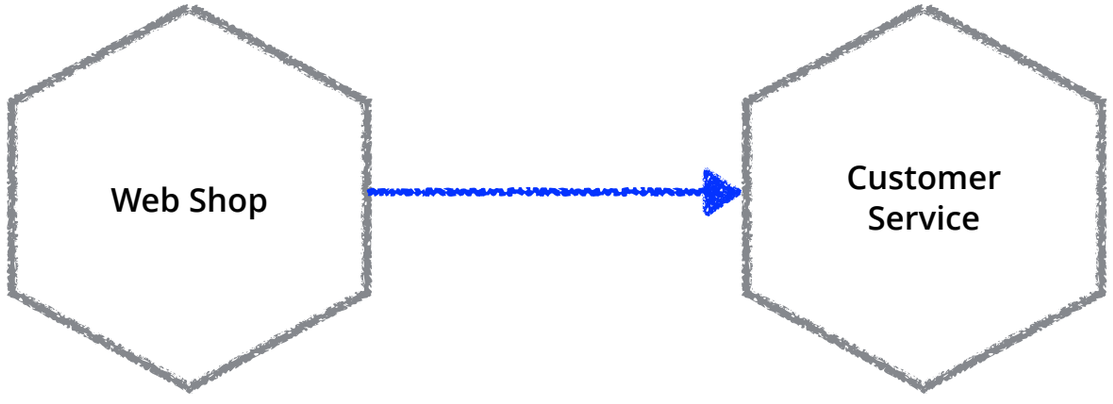


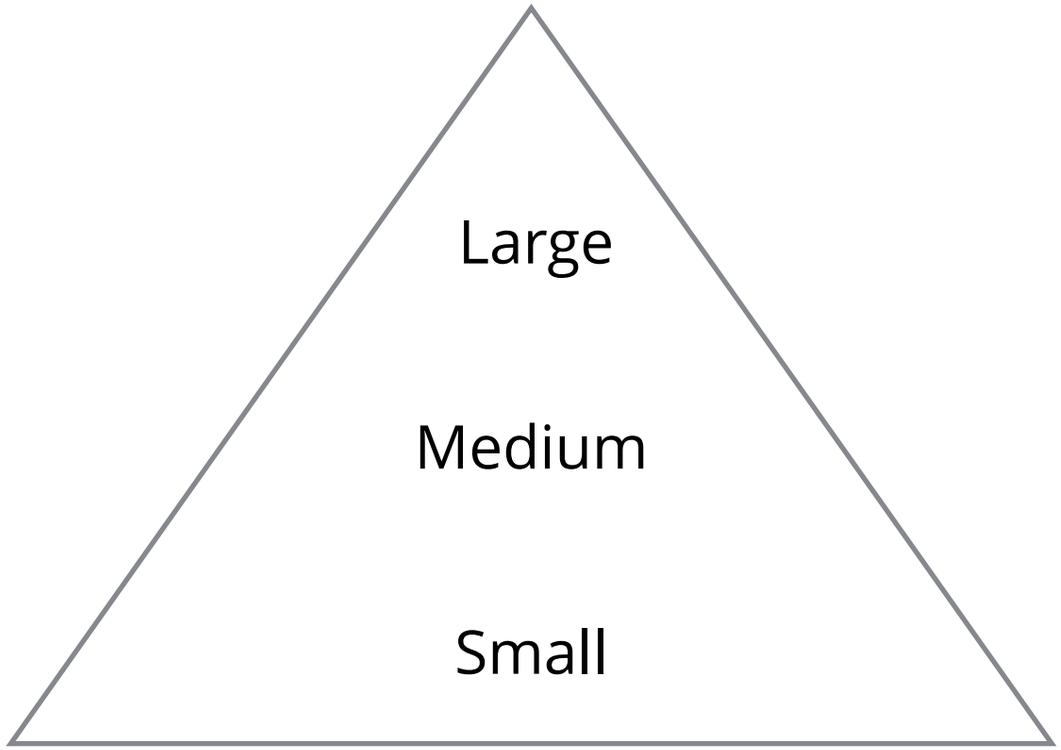


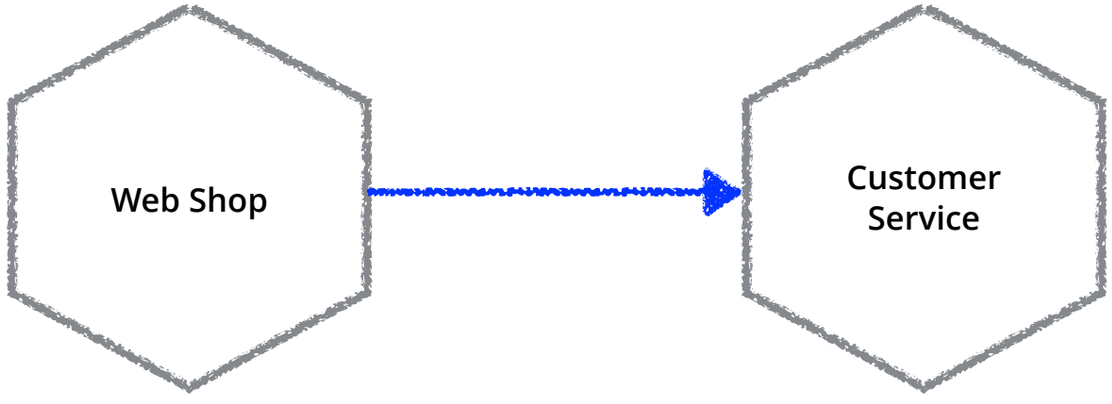
SEMANTIC MONITORING

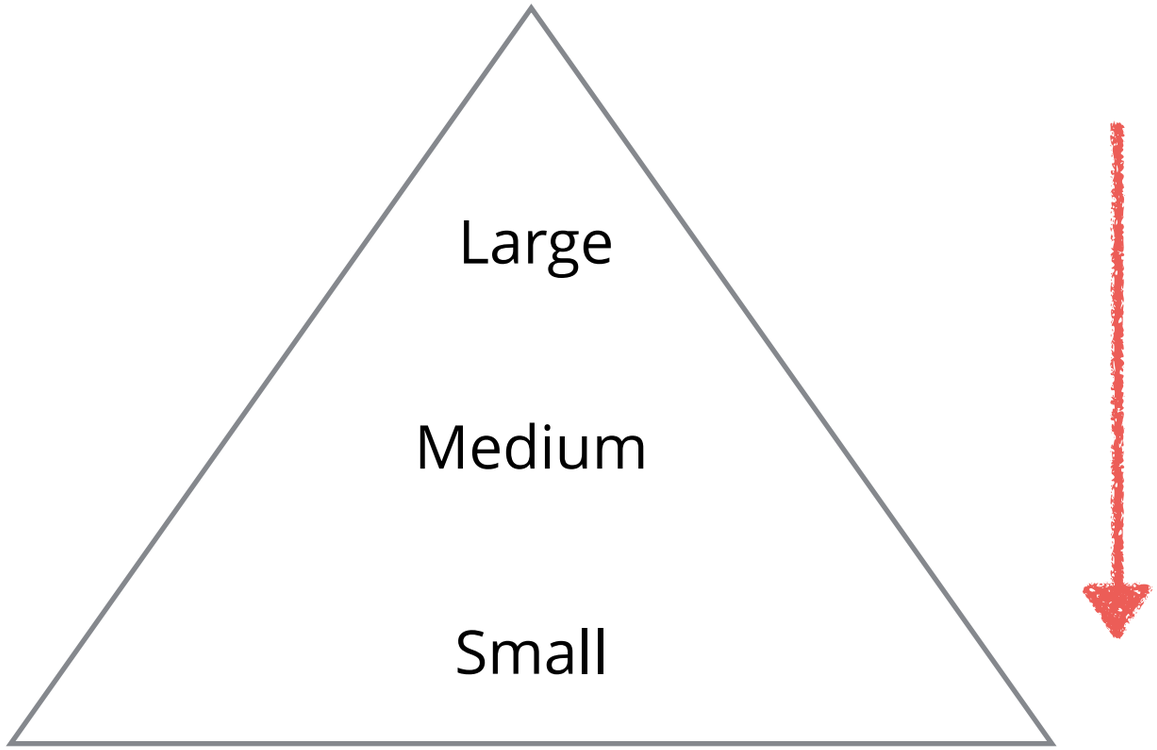


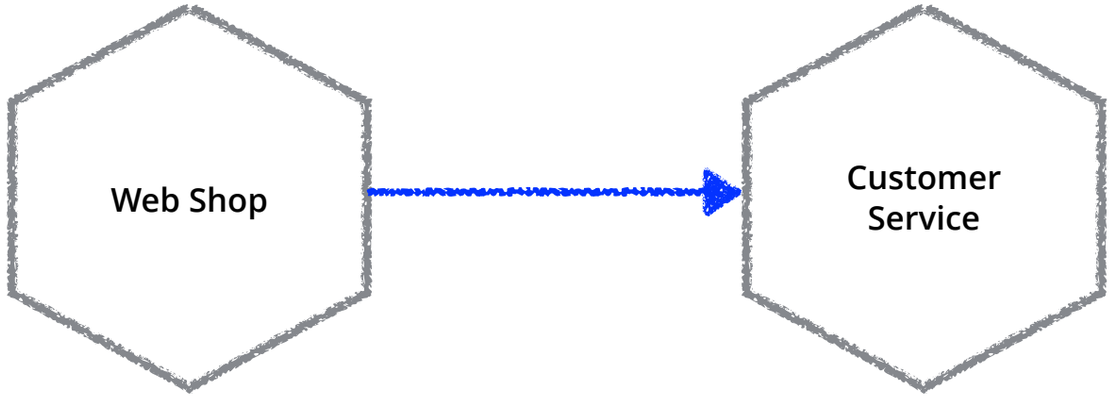

Web Shop
Customer Service


Web Shop
Customer Service

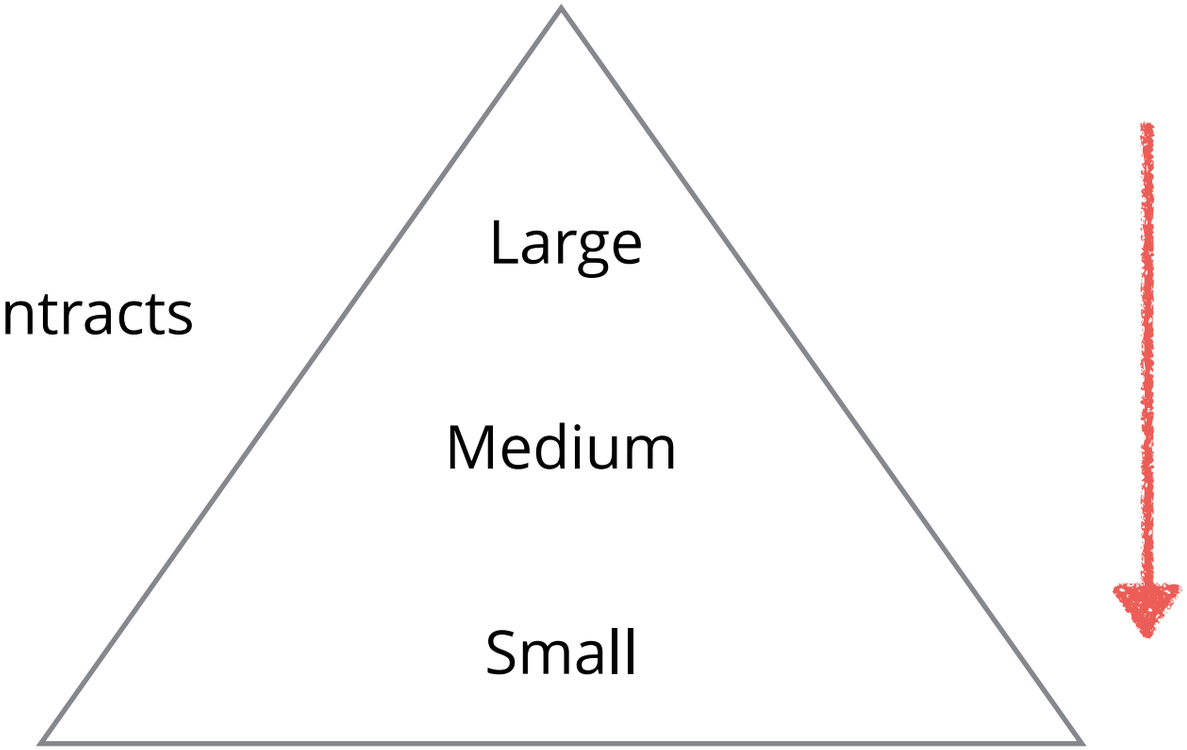

Large
Medium
Small

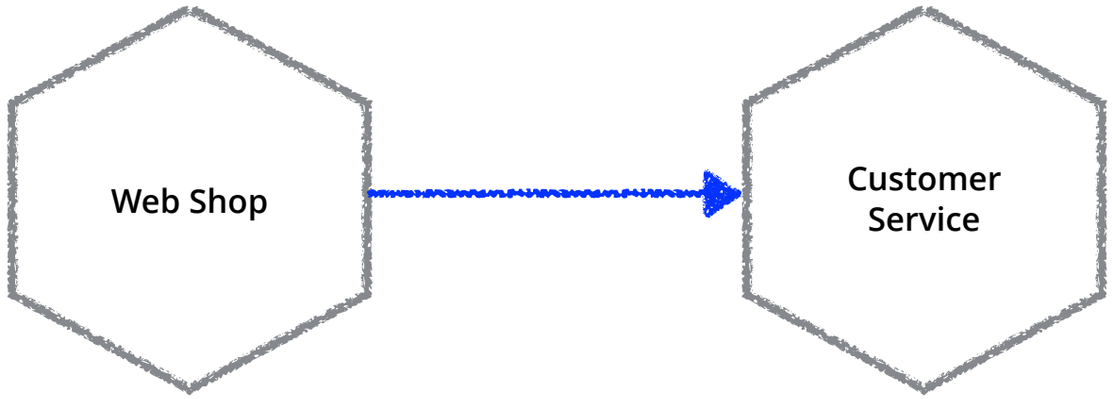

Web Shop
Customer Service

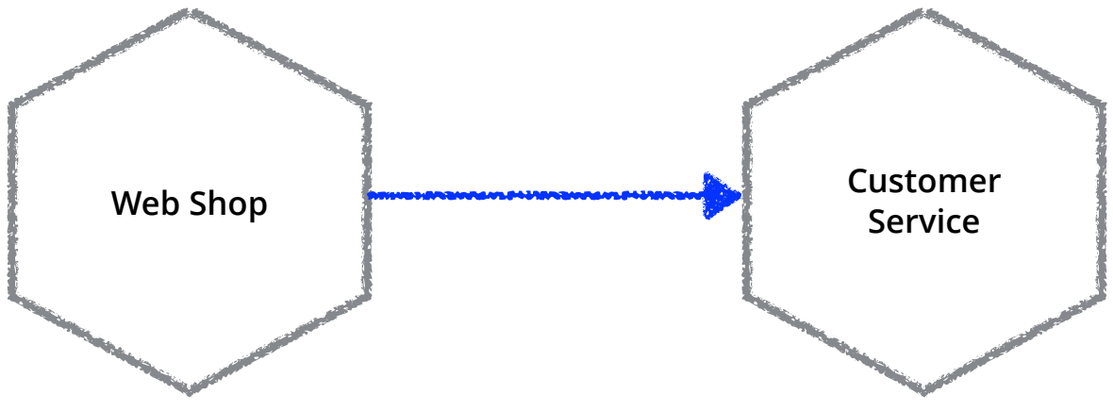

Large
Medium
Small



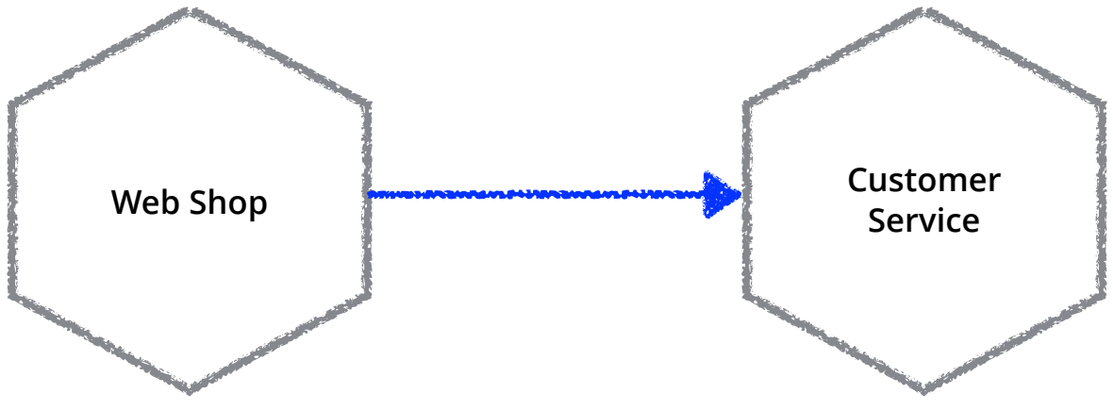
Consumer Driven Contracts




Web Shop
Customer Service

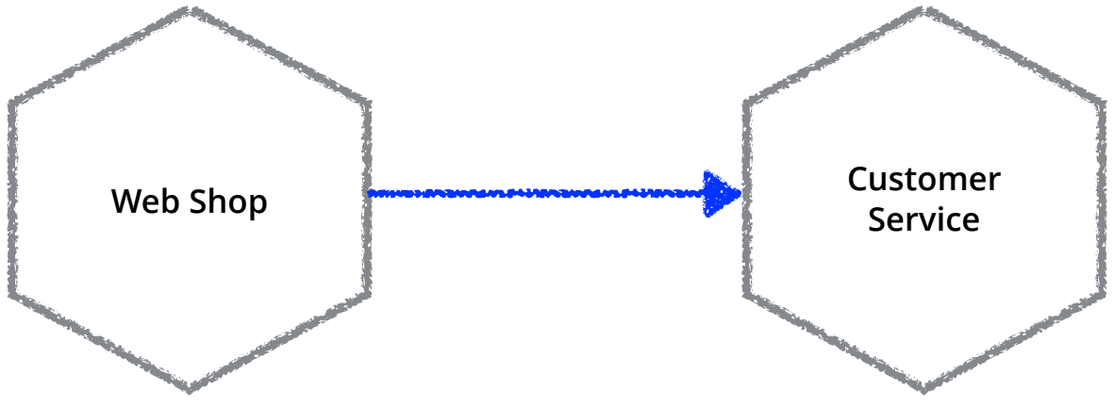


Expectations

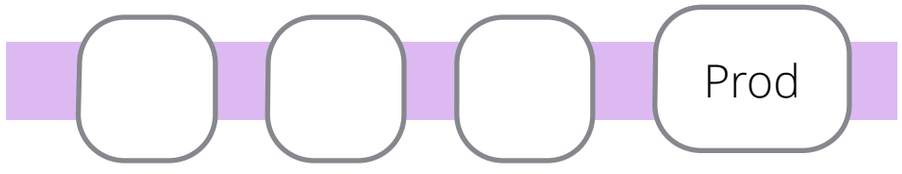


Expectations

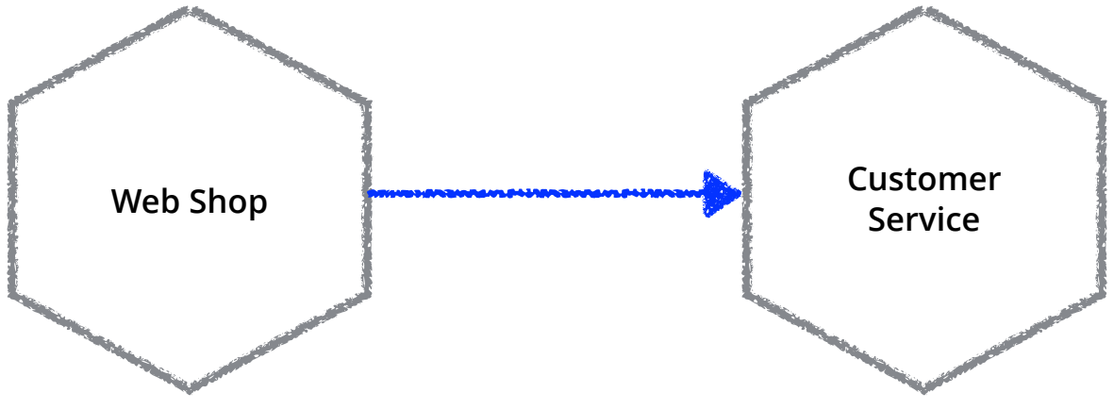
```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
            <tr valign="middle" align="center">
              <td>{ linkToBuild(build) }</td>
            </tr>
          </table>
}
```



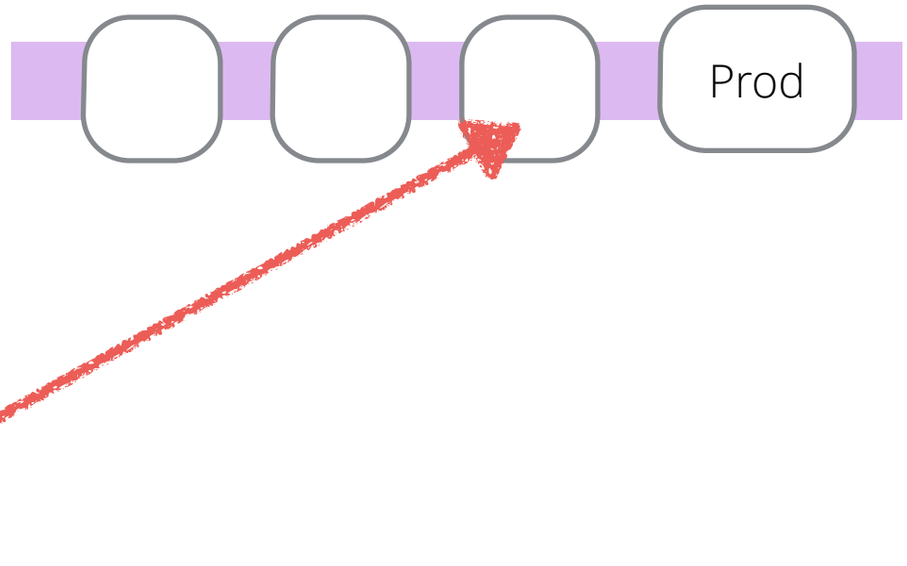
Expectations



```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
          <tr valign="middle" align="center">
            <td>{ linkToBuild(build) }</td>
          </tr>
        </table>
}
```



Expectations




```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
          <tr valign="middle" align="center">
            <td>{ linkToBuild(build) }</td>
          </tr>
        </table>
}
```

README.md

Pact

Define a pact between service consumers and providers, enabling "consumer driven contract" testing.

Pact provides an RSpec DSL for service consumers to define the HTTP requests they will make to a service provider and the HTTP responses they expect back. These expectations are used in the consumers specs to provide a mock service provider. The interactions are recorded, and played back in the service provider specs to ensure the service provider actually does provide the response the consumer expects.

This allows testing of both sides of an integration point using fast unit tests.

This gem is inspired by the concept of "Consumer driven contracts". See http://martinfowler.com/articles/consumerDrivenContracts.html for more information.

Travis CI Status: 

README.md

Pact

Define a pact between service consumers and providers, enabling "consumer driven contract" testing.

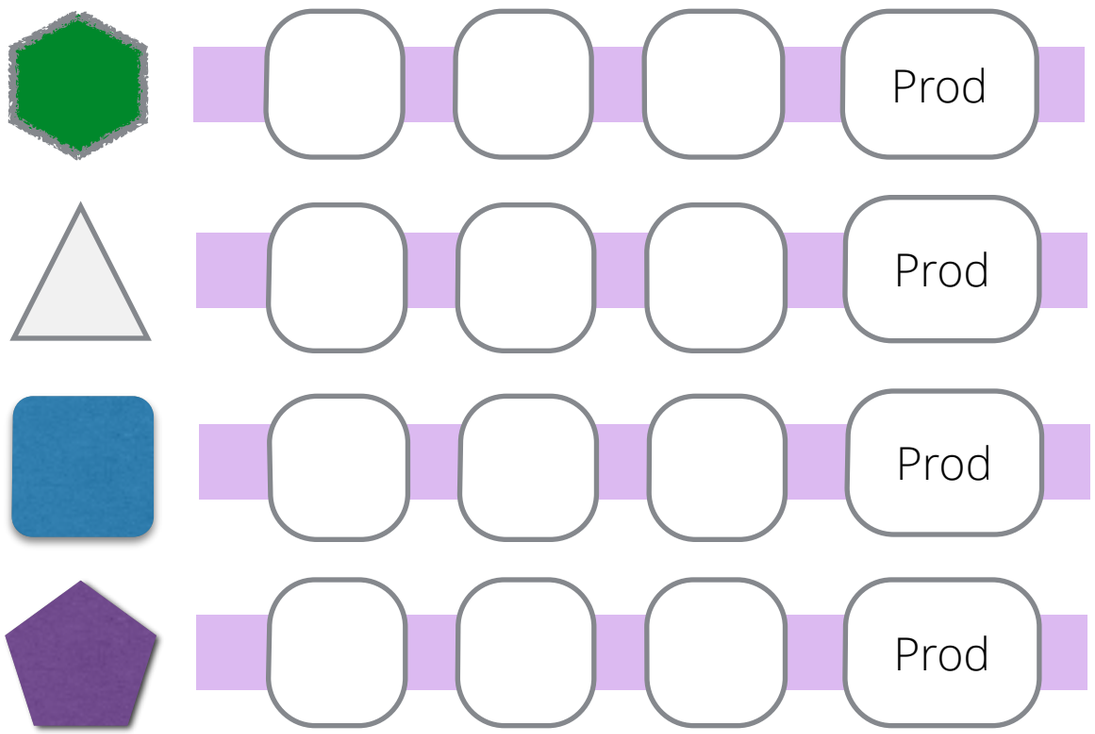
Pact provides an RSpec DSL for service consumers to define the HTTP requests they will make to a service provider and the HTTP responses they expect back. These expectations are used in the consumers specs to

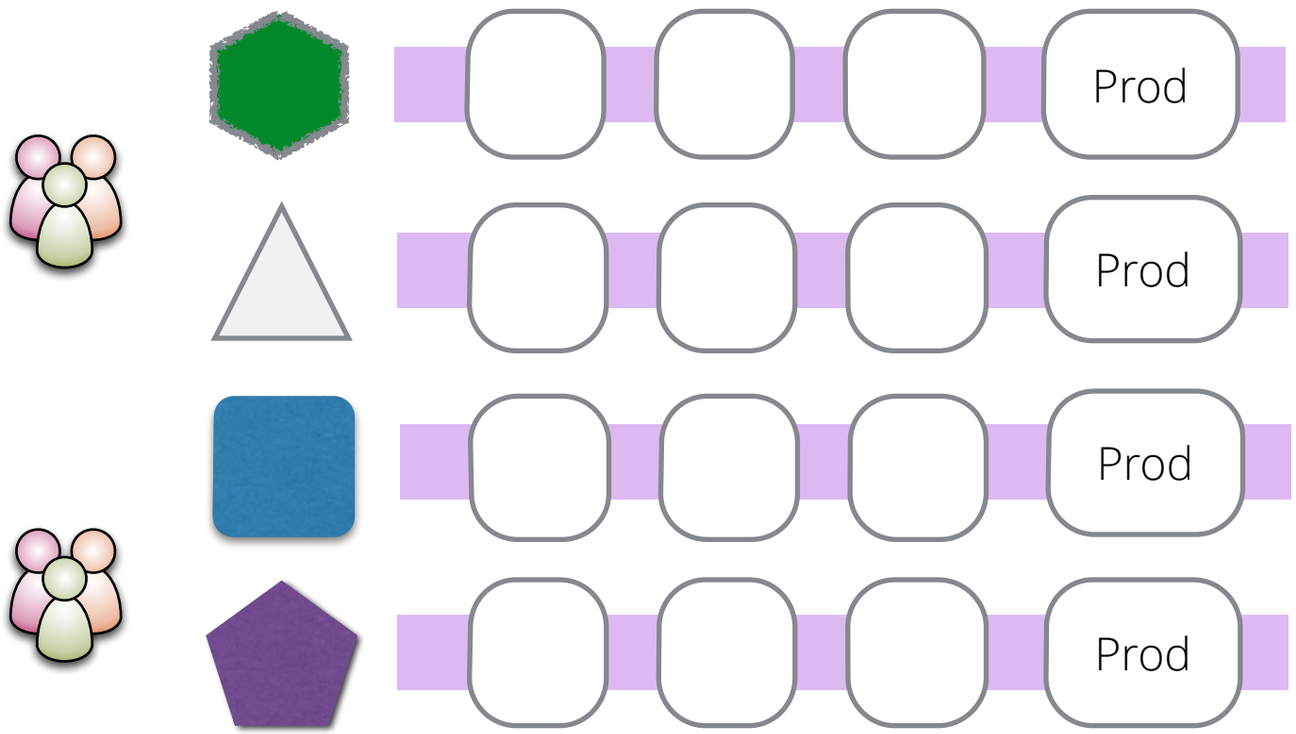
https://github.com/realestate-com-au/pact

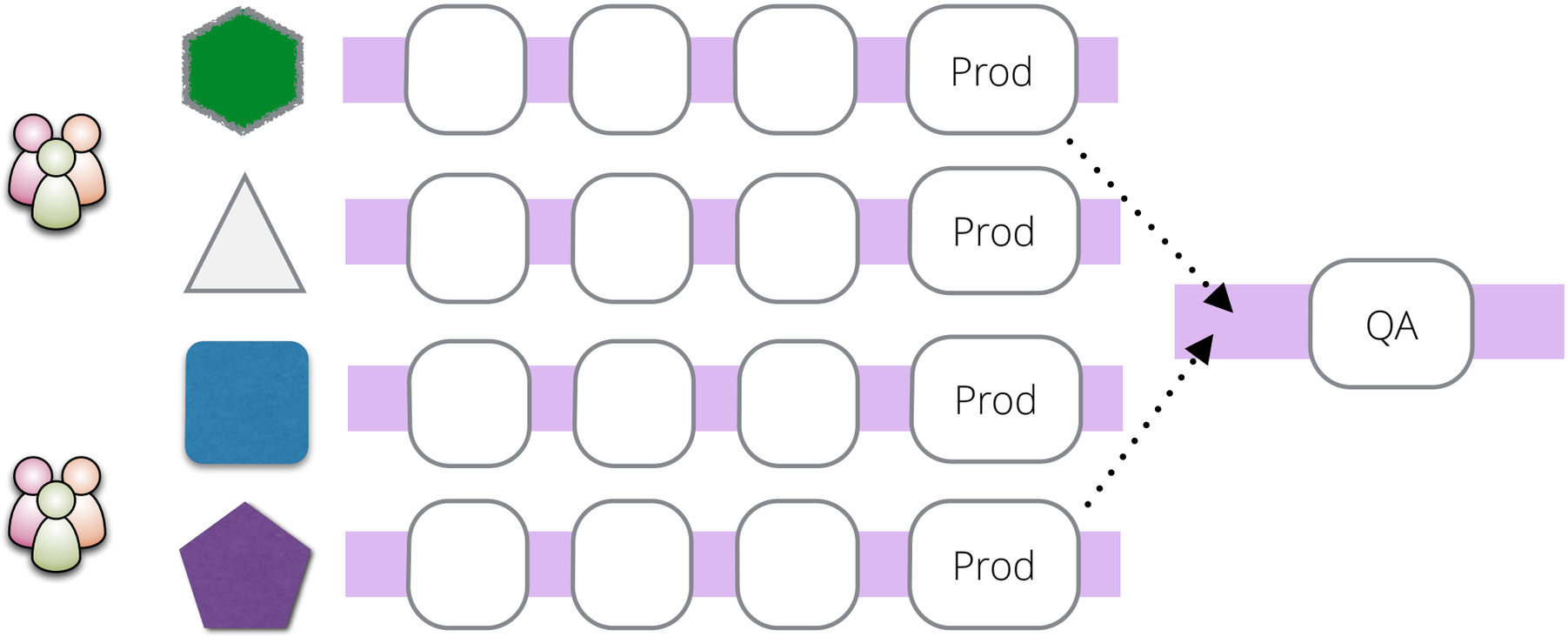
This allows testing of both sides of an integration point using fast unit tests.

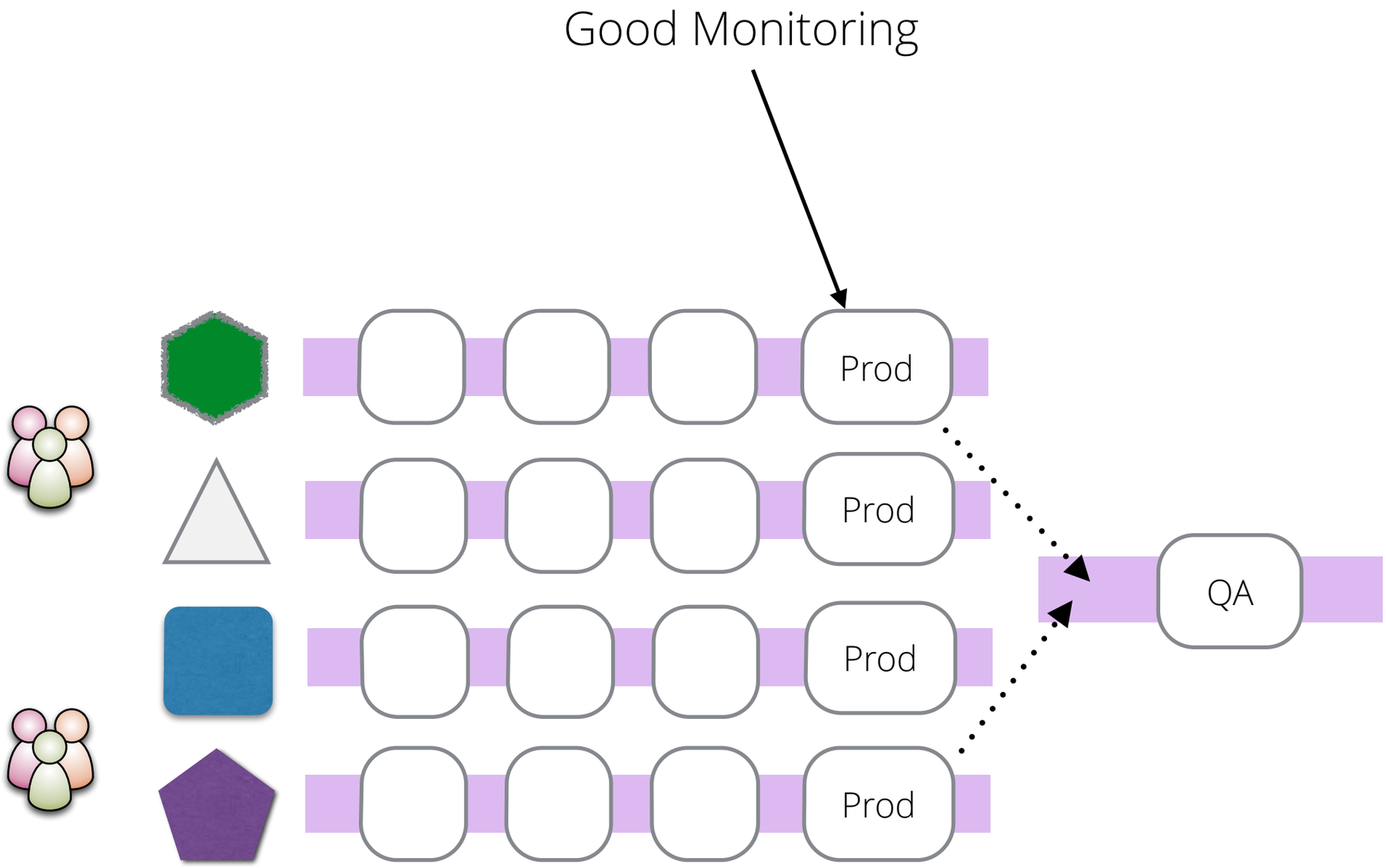
This gem is inspired by the concept of "Consumer driven contracts". See http://martinfowler.com/articles/consumerDrivenContracts.html for more information.

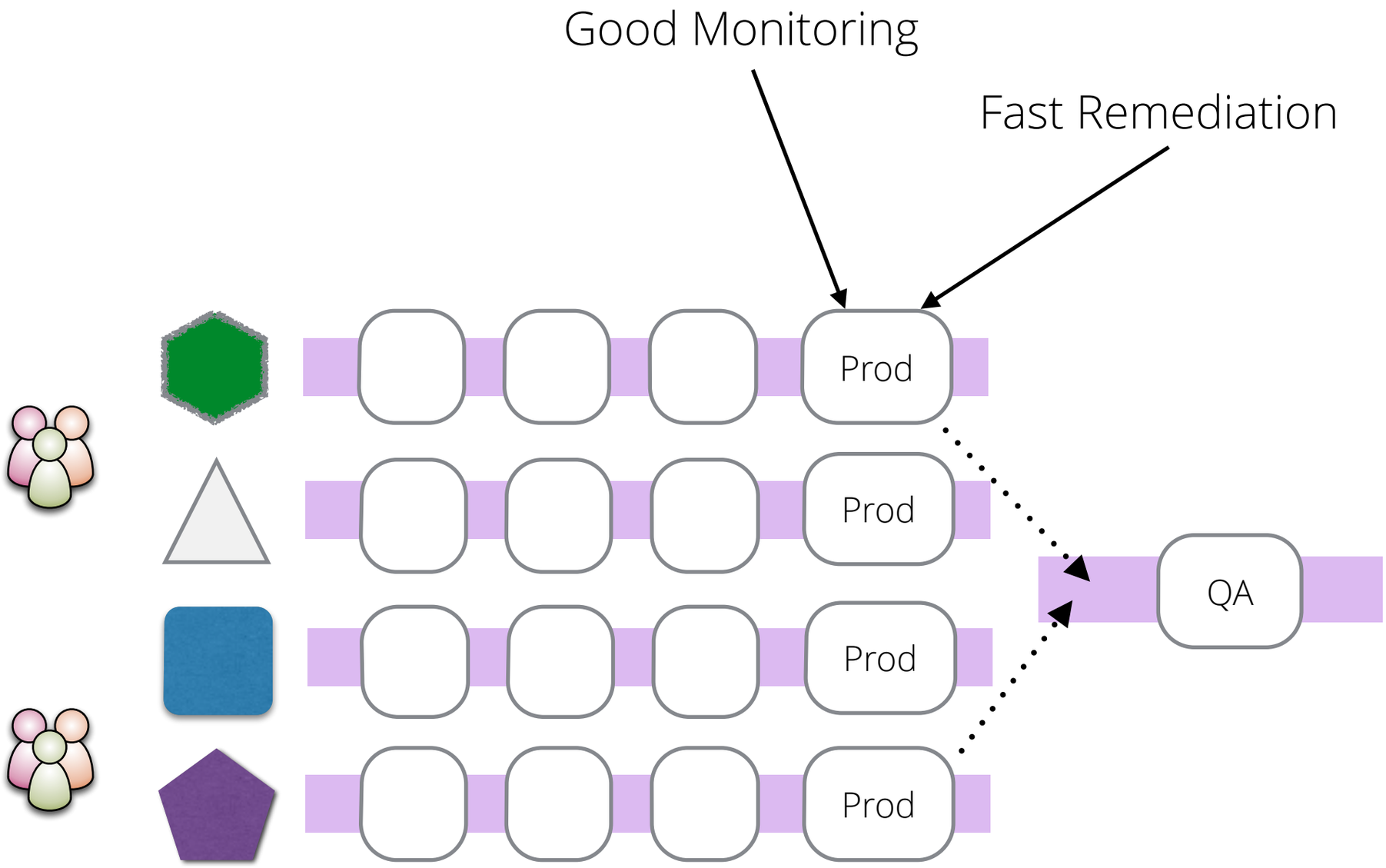
Travis CI Status: build passing

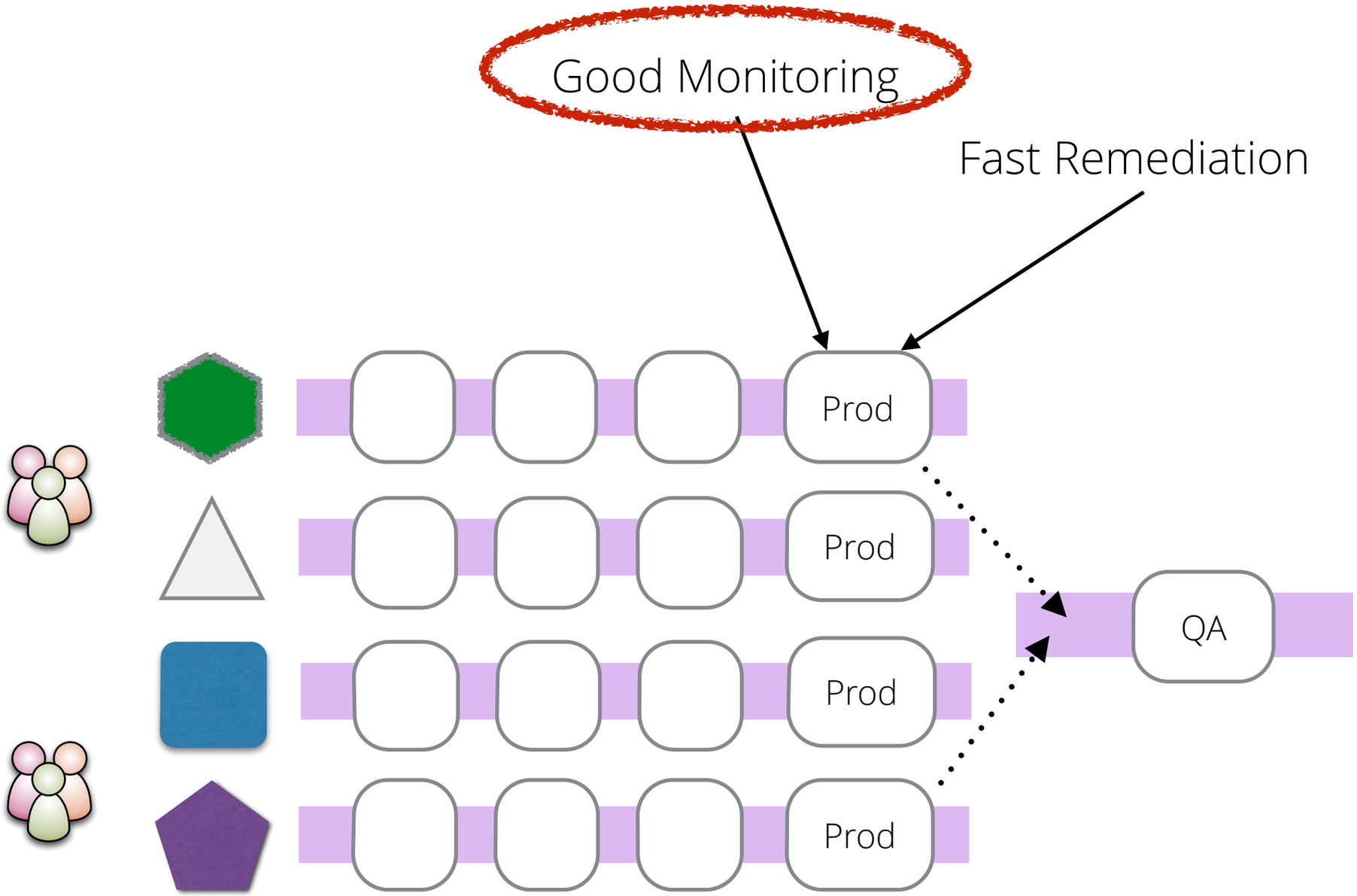

Prod
Prod
Prod
Prod


Prod
Prod
Prod
Prod


Prod
Prod
Prod
Prod
QA


Good Monitoring
Prod
Prod
Prod
Prod
QA

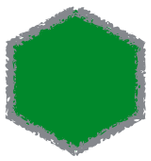

Good Monitoring
Fast Remediation
Prod
Prod
Prod
Prod
QA


Good Monitoring
Fast Remediation
Prod
Prod
Prod
Prod
QA

More Production Like

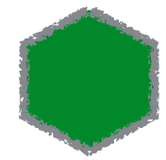
Faster Feedback

More Production Like



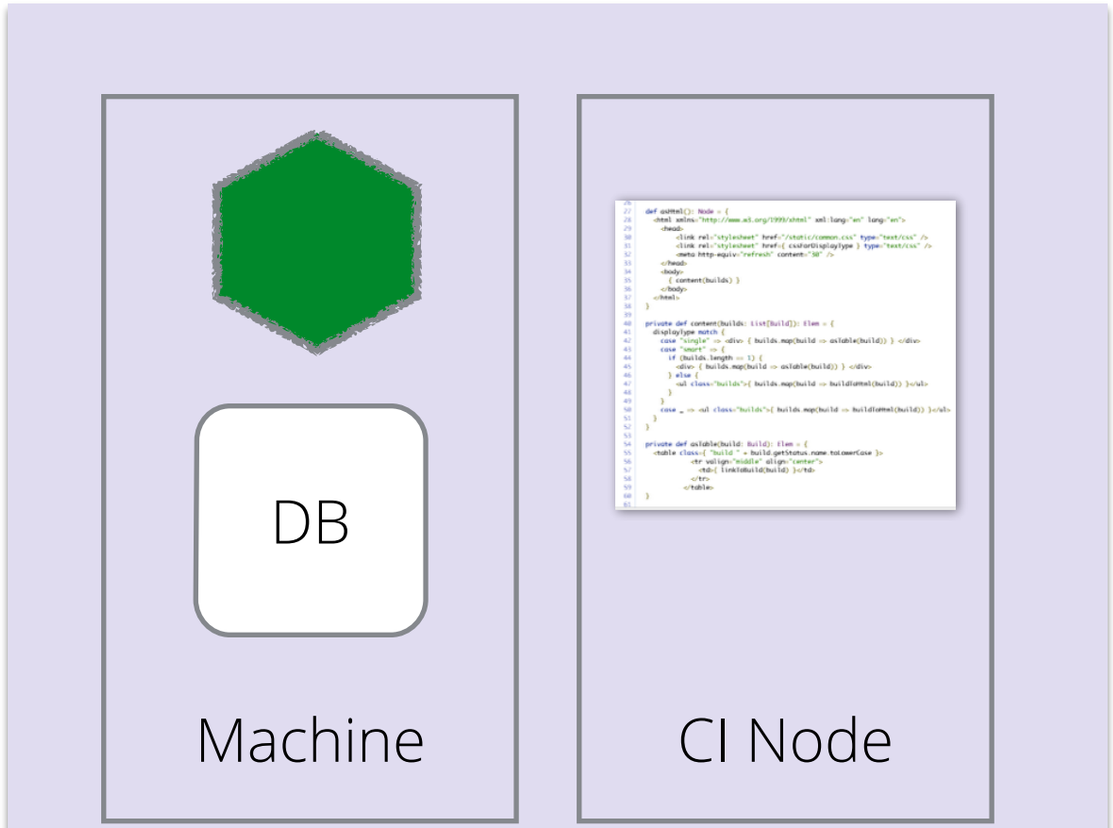
Faster Feedback

More Production Like

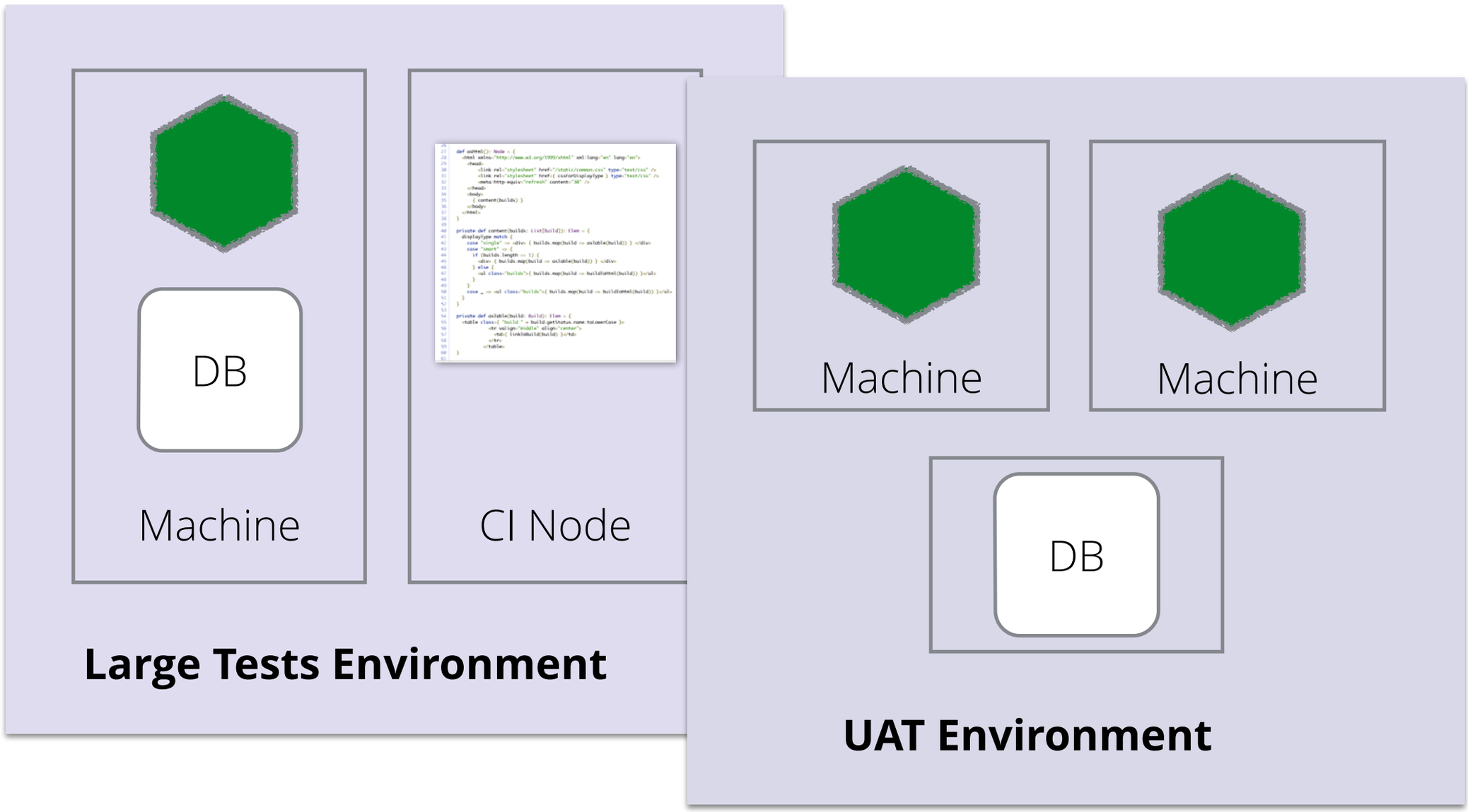


Faster Feedback

Build | S/M Tests | Large Tests | UAT | Prod



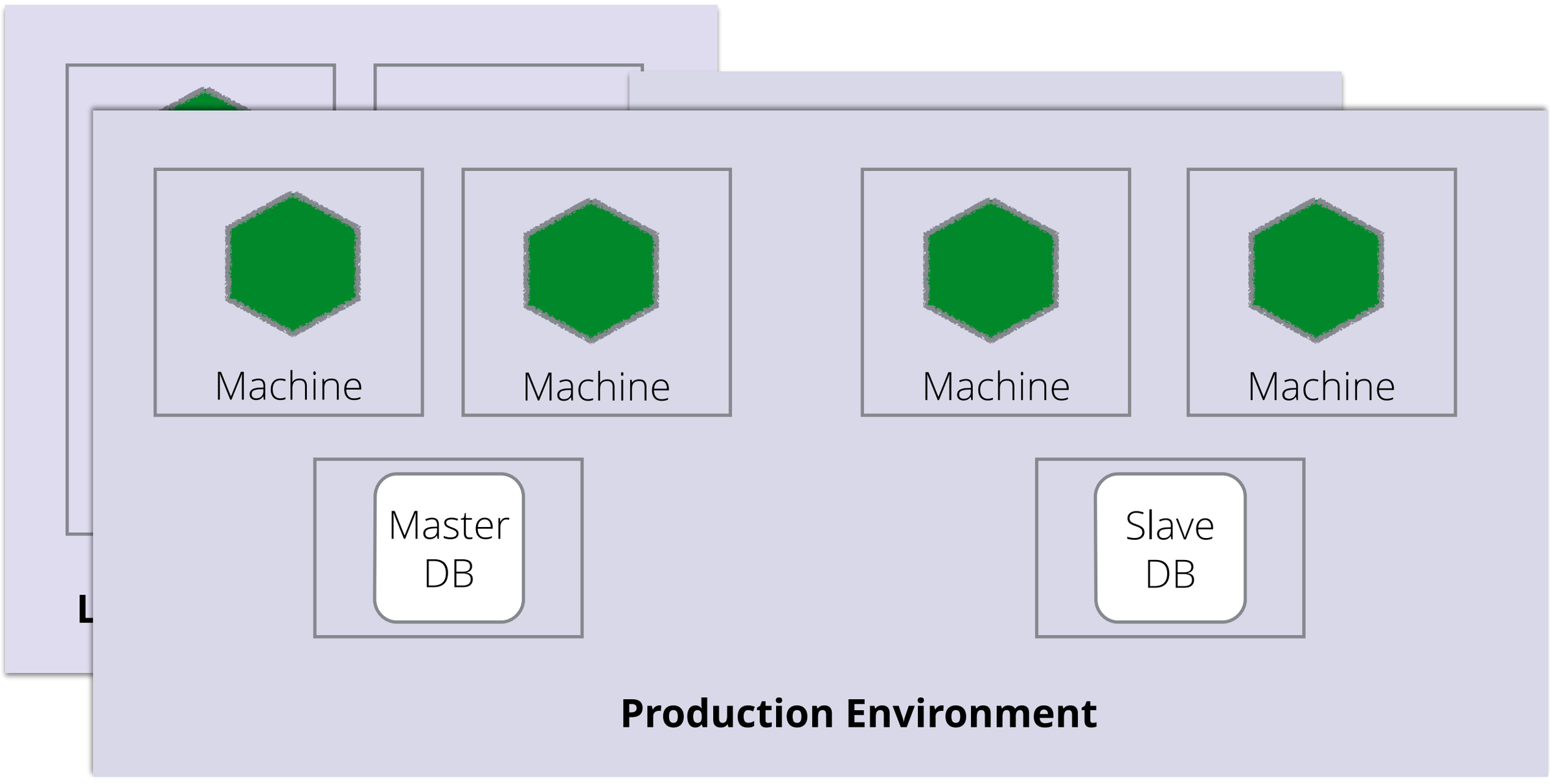
Build | S/M Tests | Large Tests | UAT | Prod



#geecon

@samnewman


Build
S/M Tests
Large Tests
UAT
Prod


Machine
Machine
Machine
Machine
Master DB
Slave DB
Production Environment

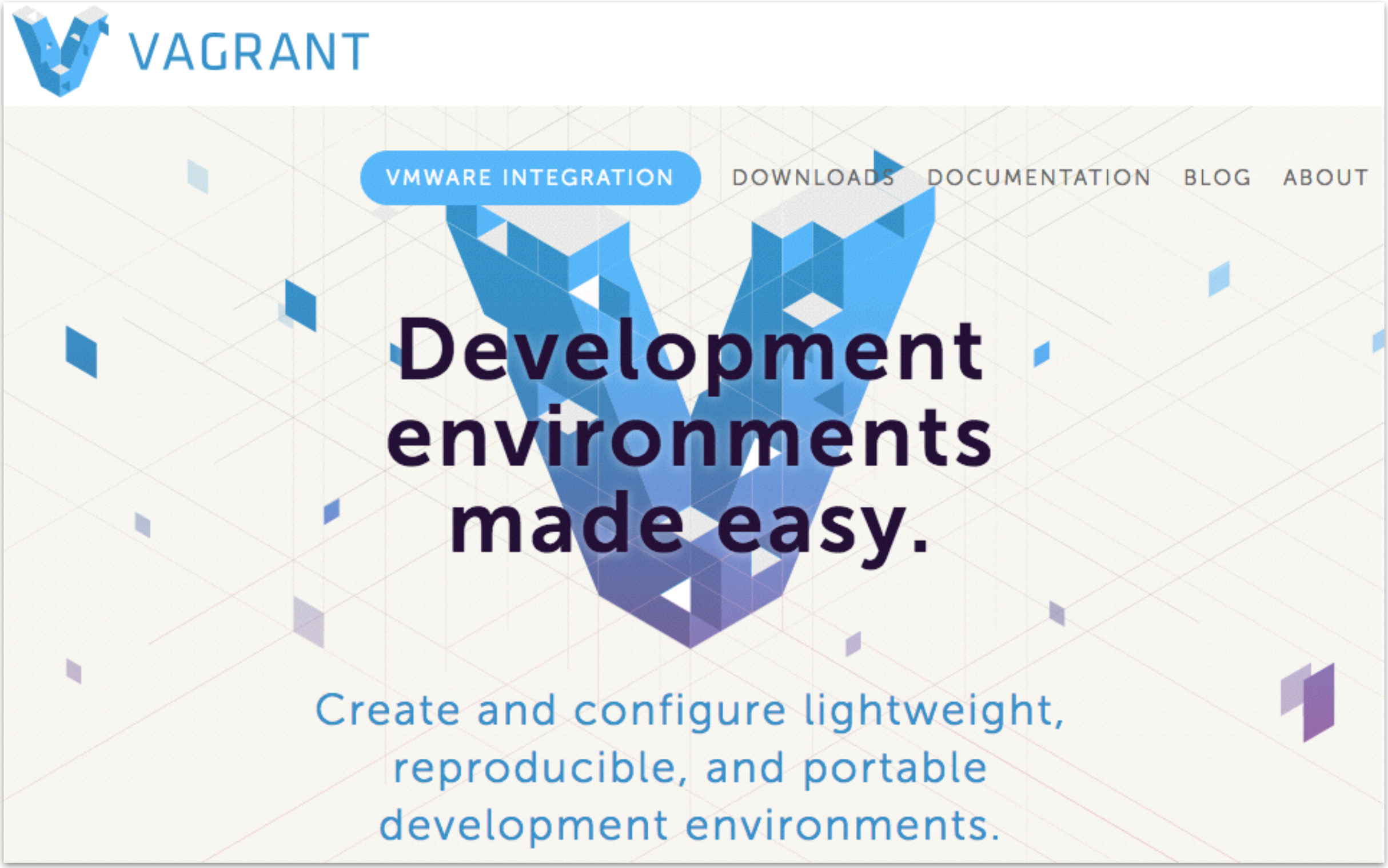
More Production Like

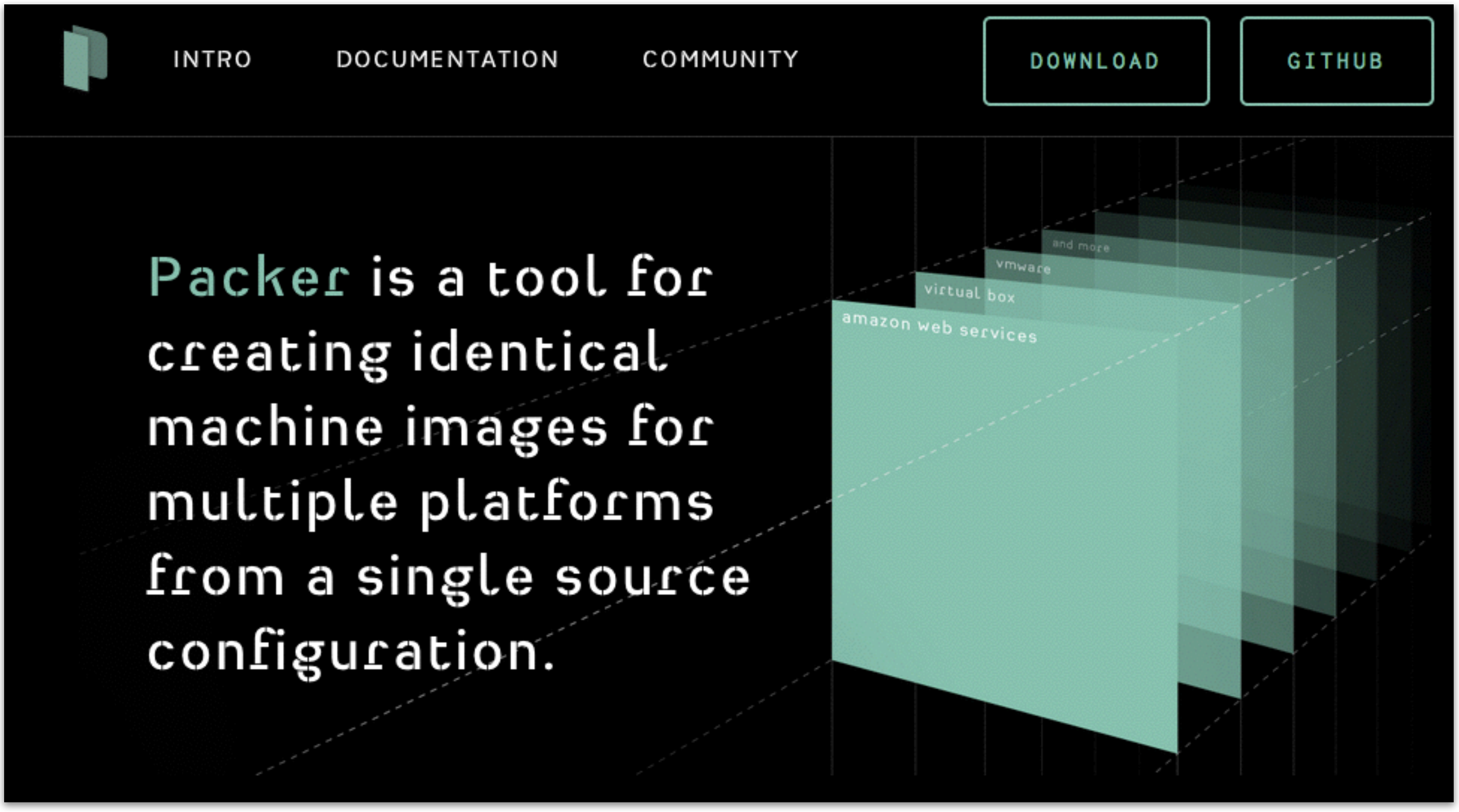
Faster Feedback

More Production Like

Build | S/M Tests | Large Tests | UAT | Prod

Faster Feedback

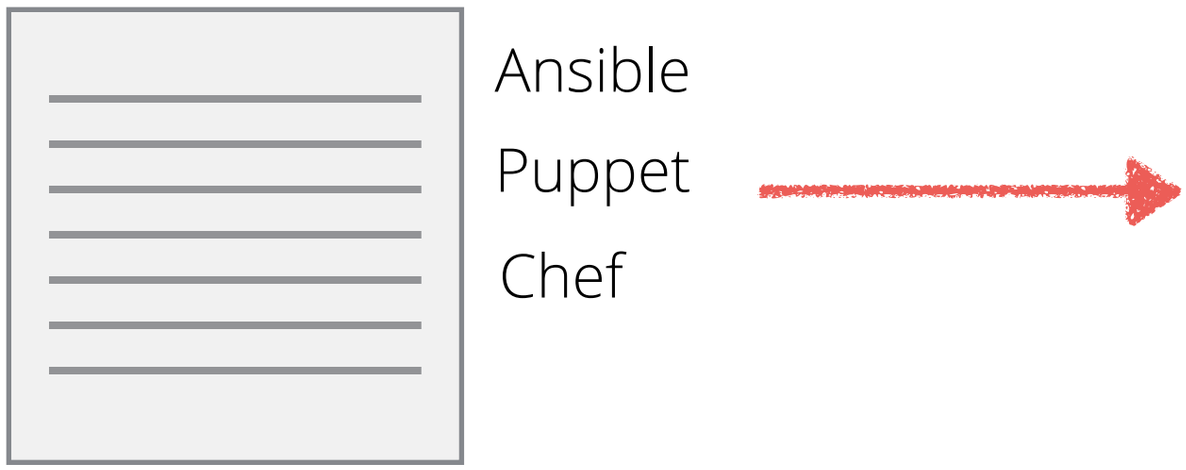

VAGRANT
VMWARE INTEGRATION
DOWNLOADS
DOCUMENTATION
BLOG
ABOUT
Development environments made easy.
Create and configure lightweight, reproducible, and portable development environments.


INTRO
DOCUMENTATION
COMMUNITY
DOWNLOAD
GITHUB
Packer is a tool for creating identical machine images for multiple platforms from a single source configuration.
and more
vmware
virtual box
amazon web services

Ansible

Puppet

Chef


Ansible
Puppet
Chef

AWS

Ansible

Puppet

Chef

AWS

Digital Ocean

Ansible

Puppet

Chef

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Vagrant

Immutable Servers

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Vagrant

Immutable Servers

Fast Spin-up

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Vagrant

Immutable Servers

Fast Spin-up

Provider Agnostic

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Vagrant

Immutable Servers

Fast Spin-up

Provider Agnostic

Ansible

Puppet

Chef

AWS

Digital Ocean

OpenStack

VMWare

Vagrant

Feedback Can Suffer

Immutable Servers

Fast Spin-up

Provider Agnostic

Ansible

Puppet

Chef

AWS

Digital Ocean

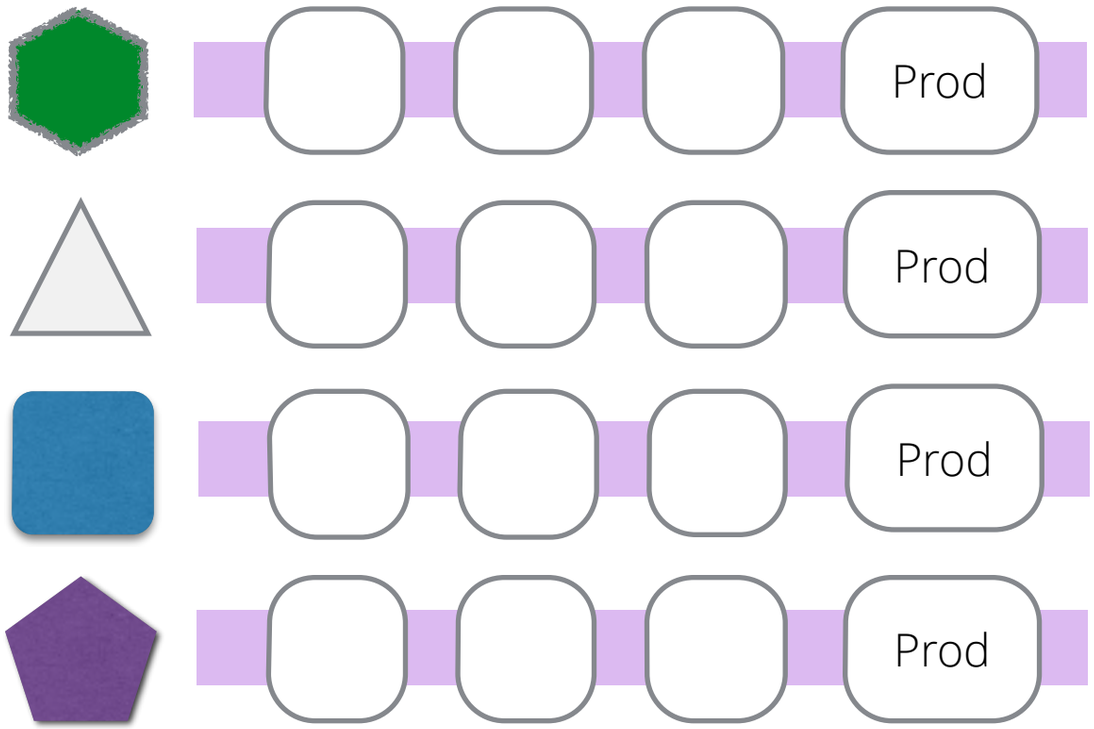
OpenStack

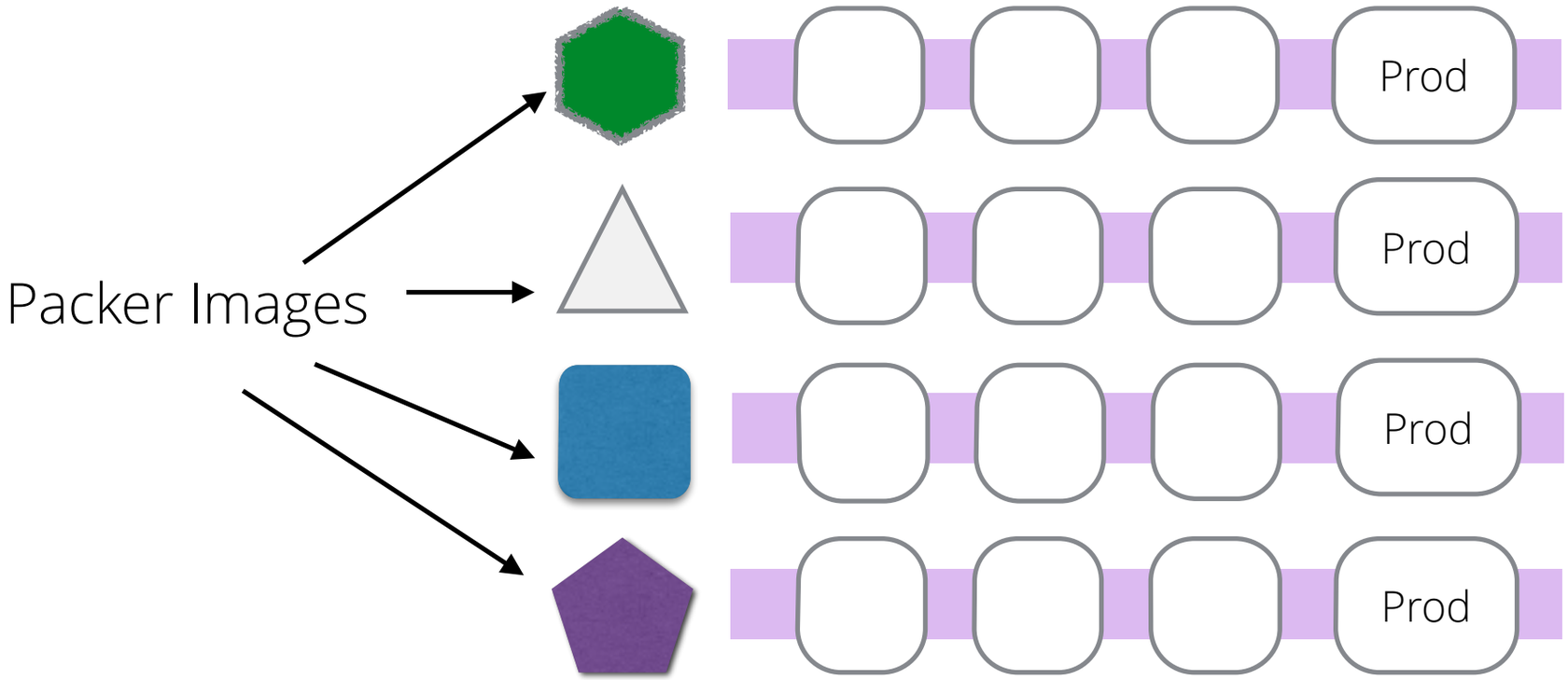
VMWare

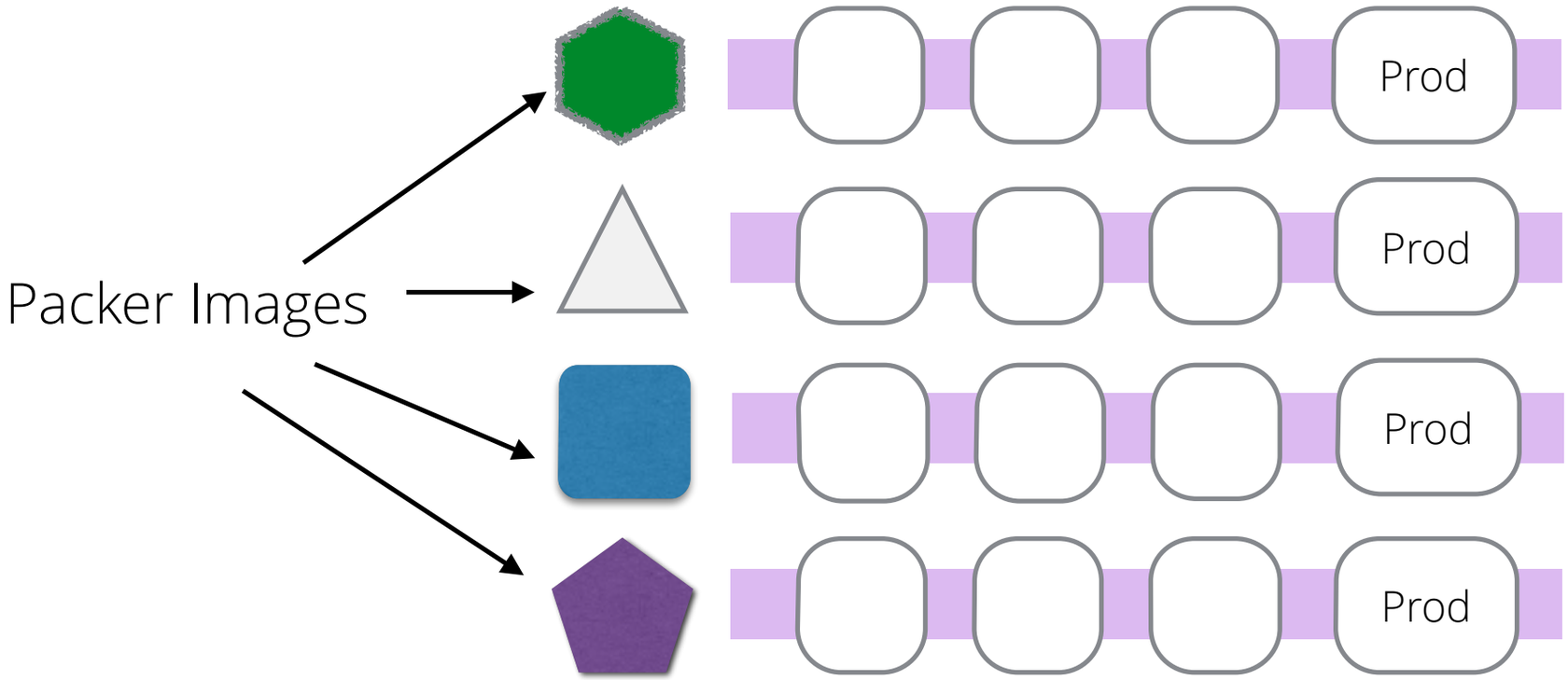
Vagrant

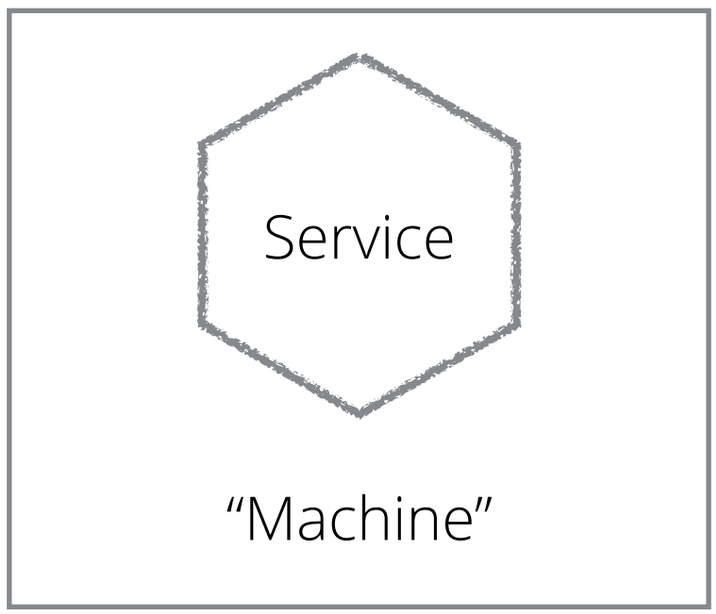
Feedback Can Suffer

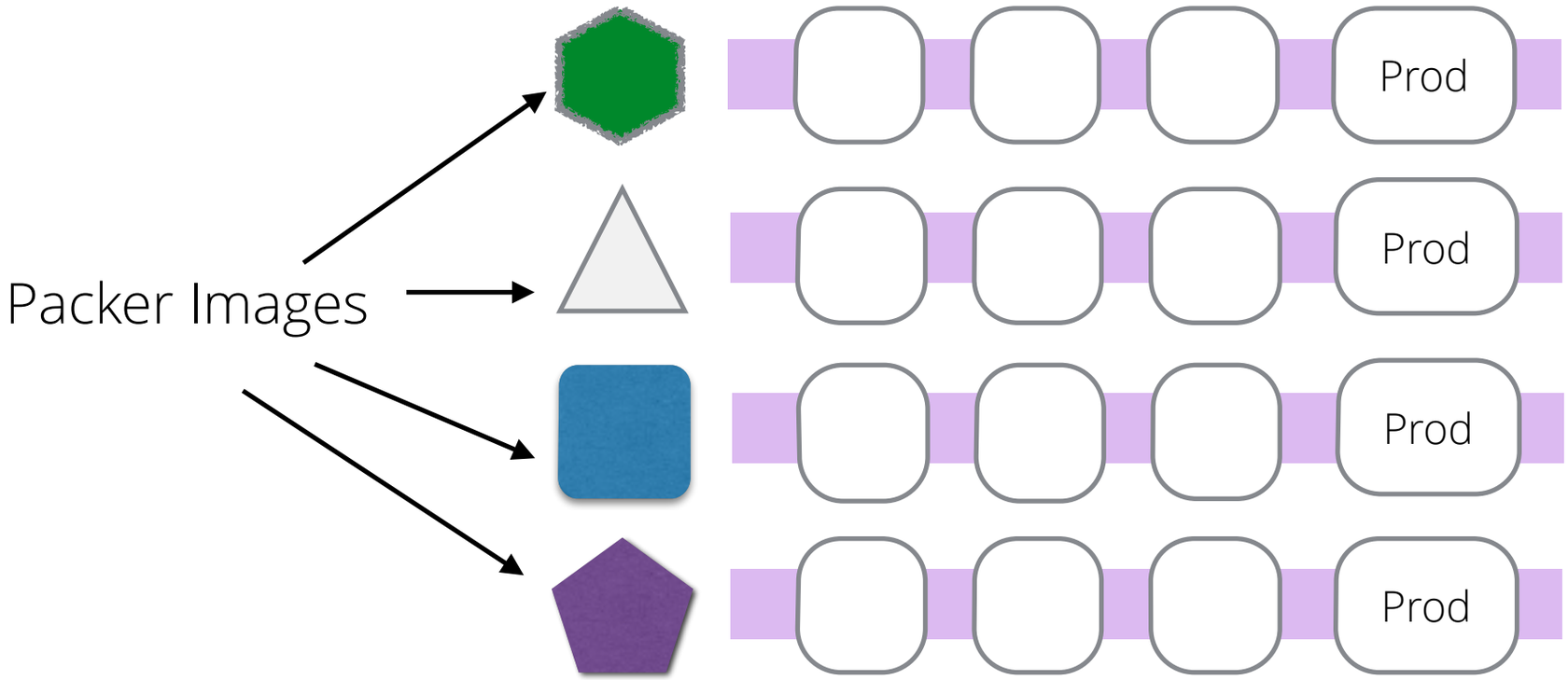
Cycle Time

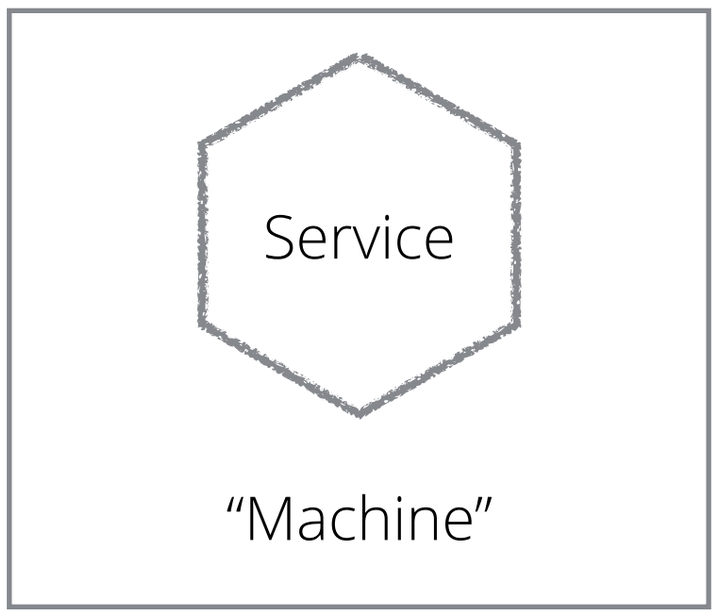

Prod
Prod
Prod
Prod

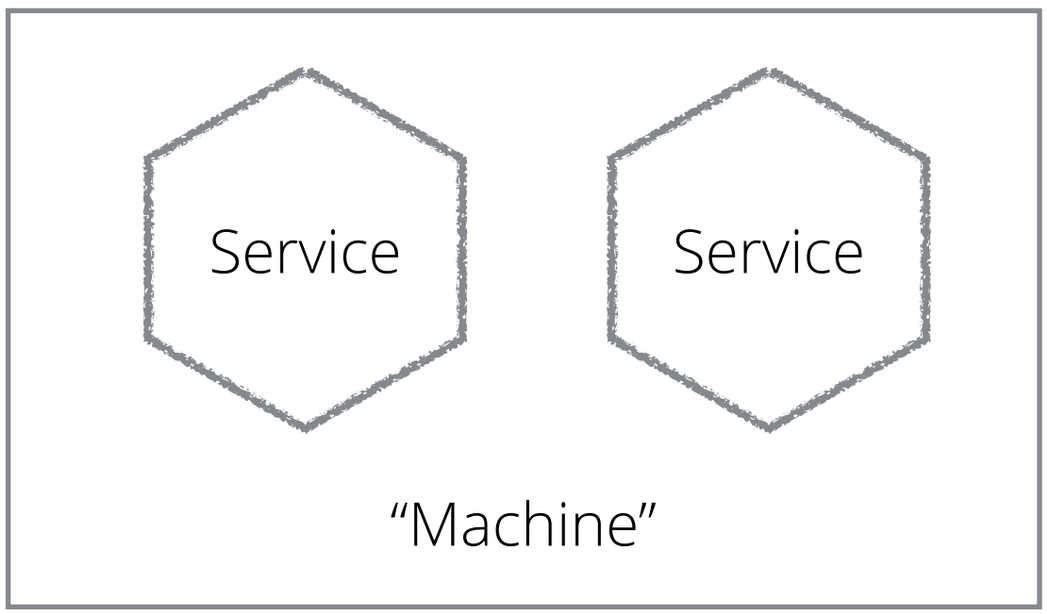

Packer Images
Prod
Prod
Prod
Prod

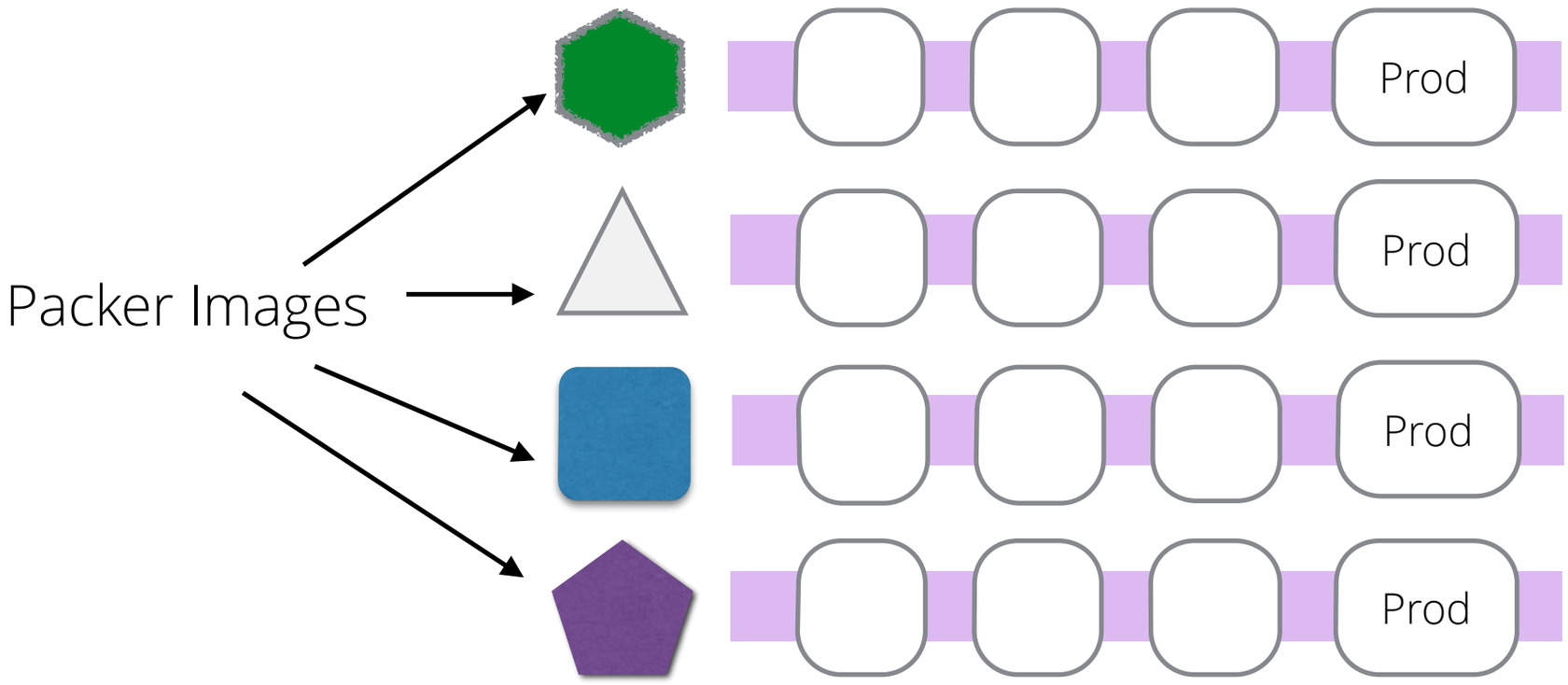

Packer Images
Prod
Prod
Prod
Prod

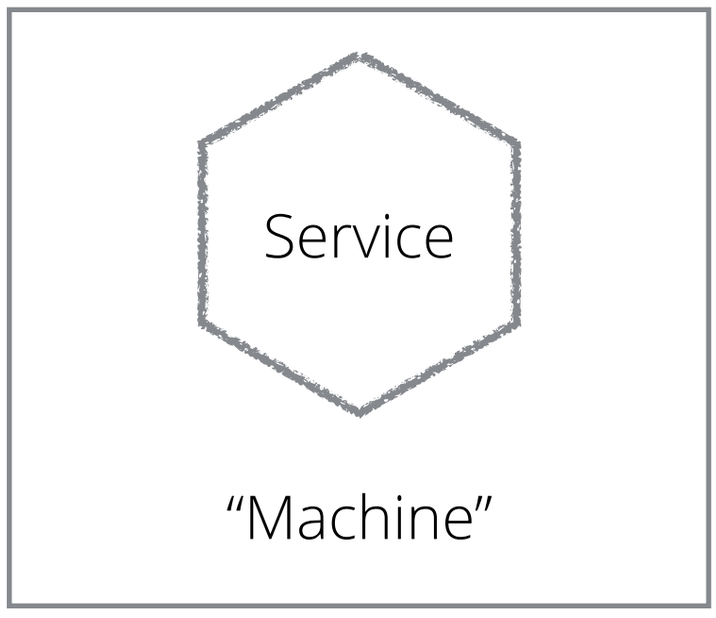

Service
“Machine”

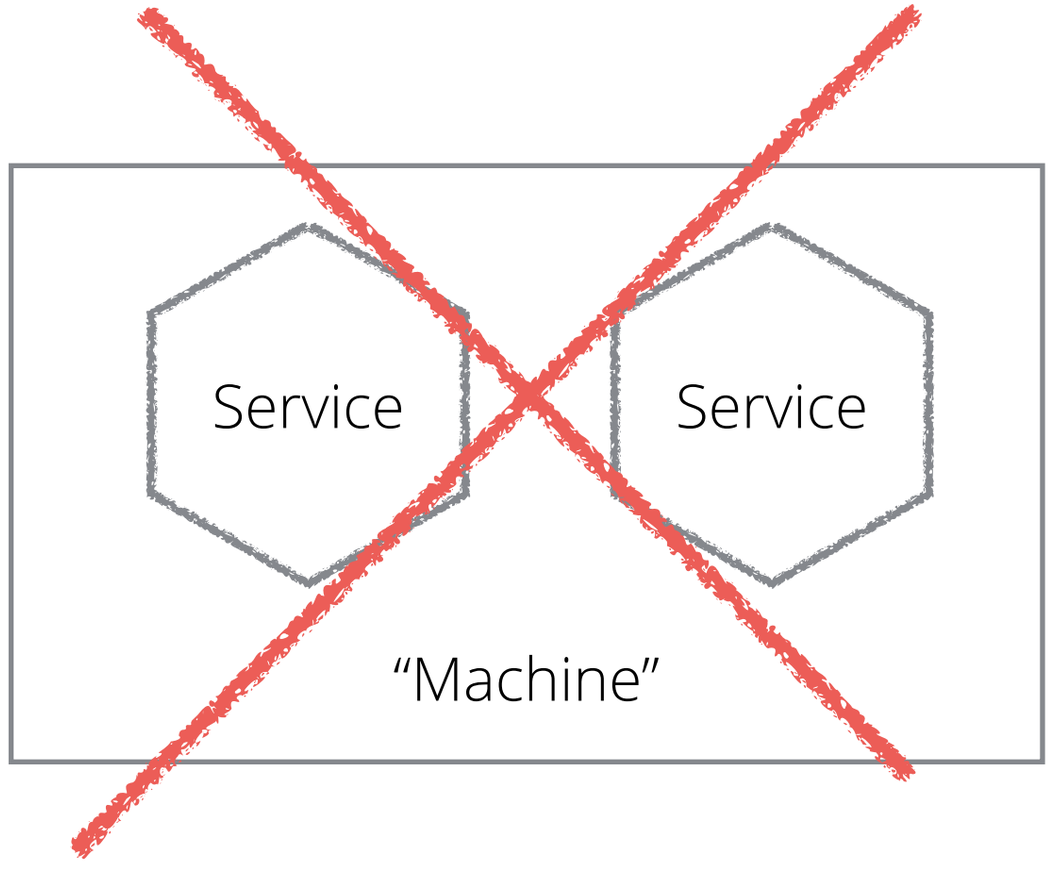

Packer Images
Prod
Prod
Prod
Prod


Service
"Machine"


Service
Service
"Machine"


Packer Images
Prod
Prod
Prod
Prod


Service
“Machine”


Service
Service
“Machine”


Build
S/M Tests
Large Tests
UAT
Prod

AWS

AWS

VMWare

AWS

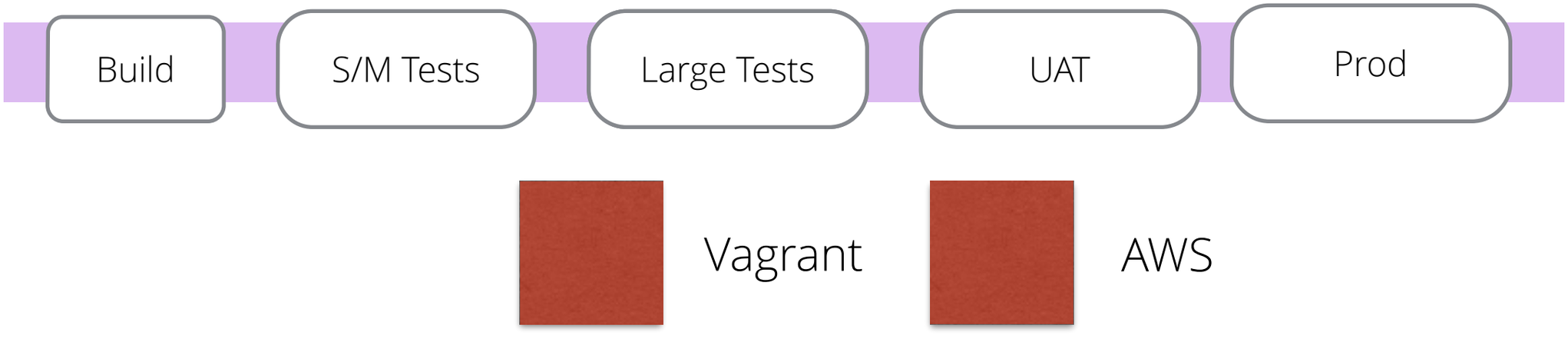
VMWare

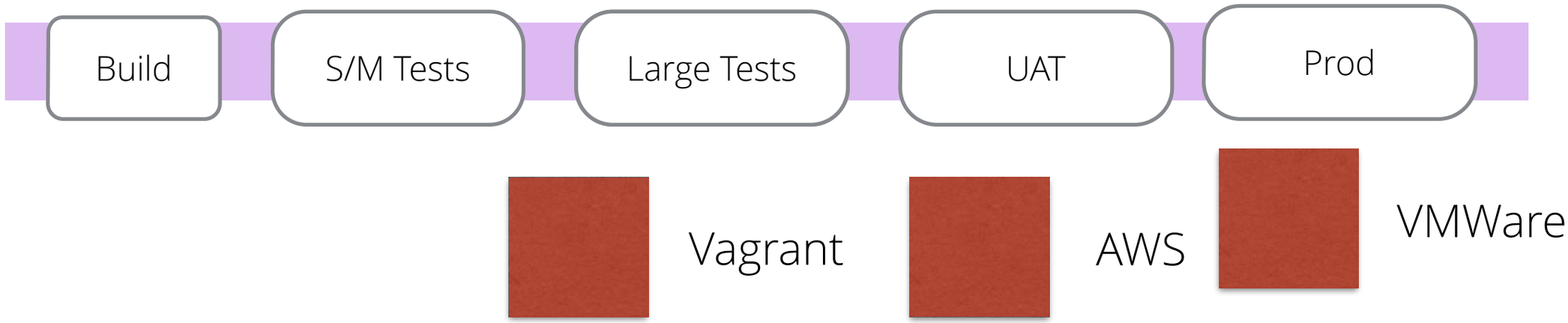
Vagrant

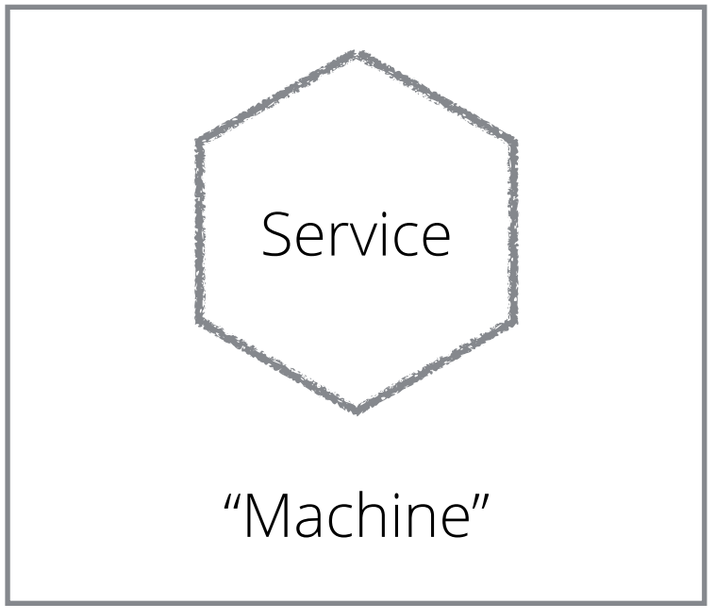
Vagrant

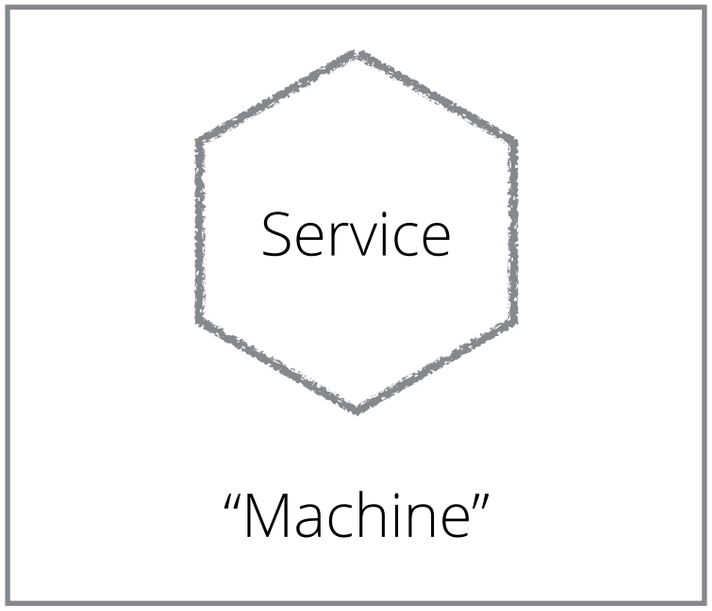
AWS

VMWare

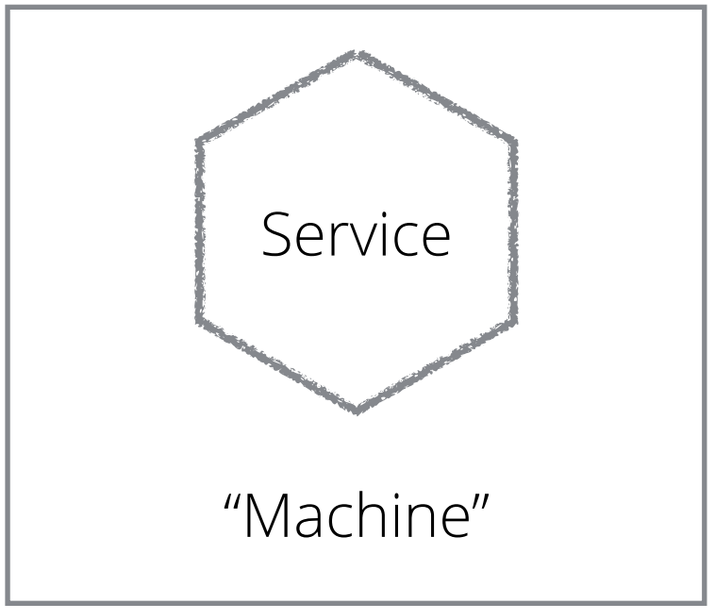

Build
S/M Tests
Large Tests
UAT
Prod
Vagrant
AWS
VMWare


Build
S/M Tests
Large Tests
UAT
Prod
Vagrant
AWS
VMWare


Service
"Machine"

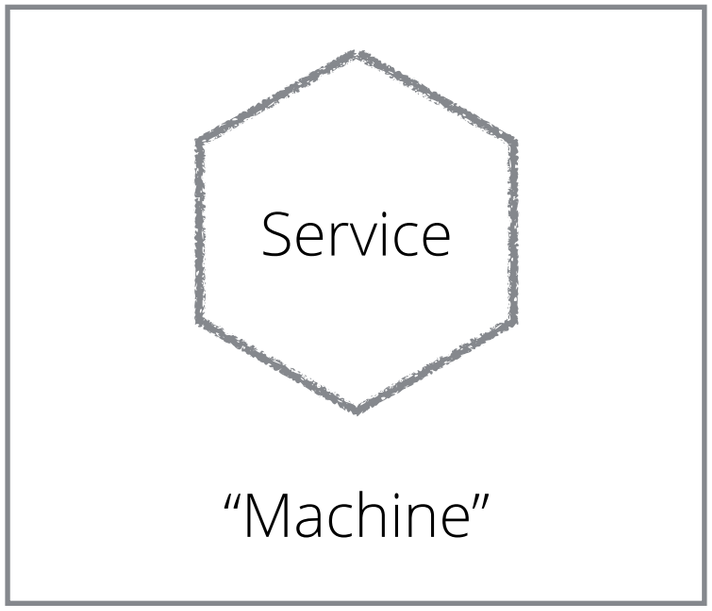


Much Easier To Reason About



Much Easier To Reason About

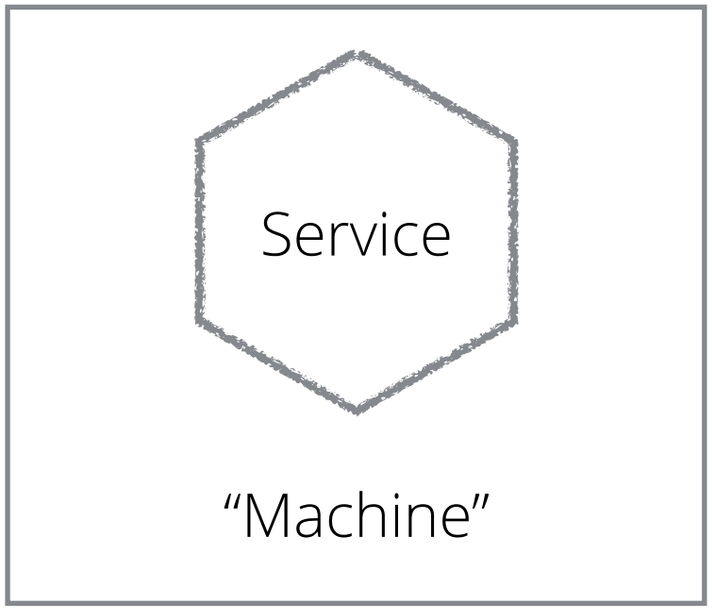
Easier To Provision (Or Decommission)



Much Easier To Reason About

Easier To Provision (Or Decommission)

Fewer Side-effects

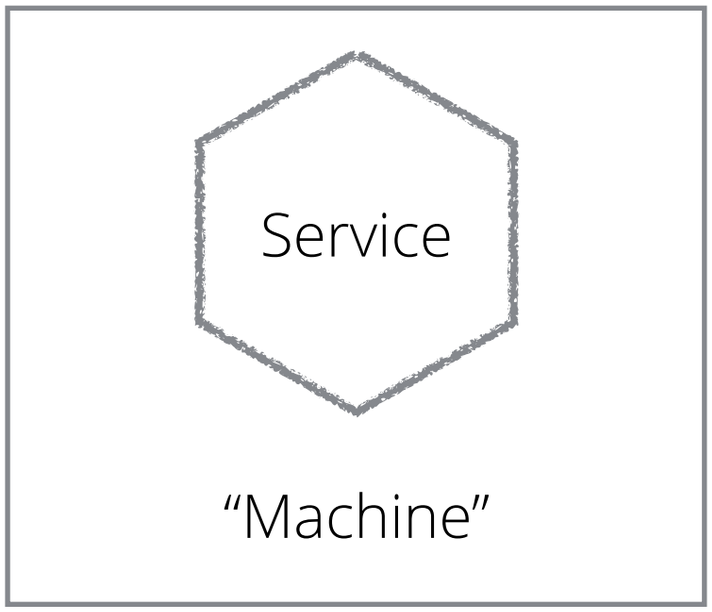


Much Easier To Reason About

Easier To Provision (Or Decommission)

Fewer Side-effects

Cost & Management Overhead!



Much Easier To Reason About

Easier To Provision (Or Decommission)

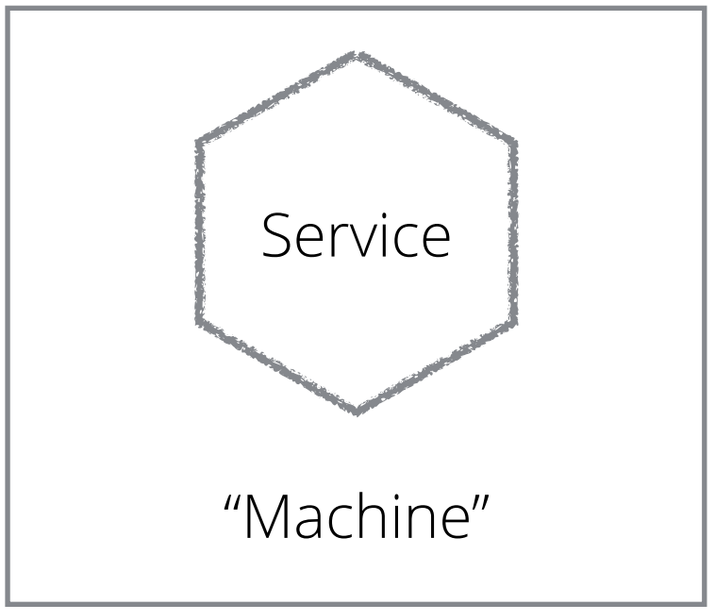
Fewer Side-effects

Cost & Management Overhead!

AWS

Digital Ocean

OpenStack



Much Easier To Reason About

Easier To Provision (Or Decommission)

Fewer Side-effects

Cost & Management Overhead!

AWS

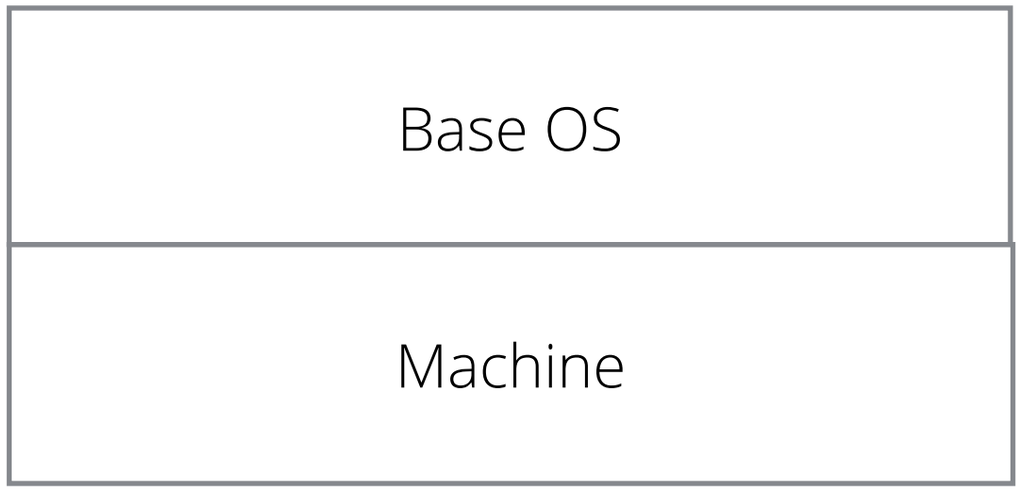
Digital Ocean

OpenStack

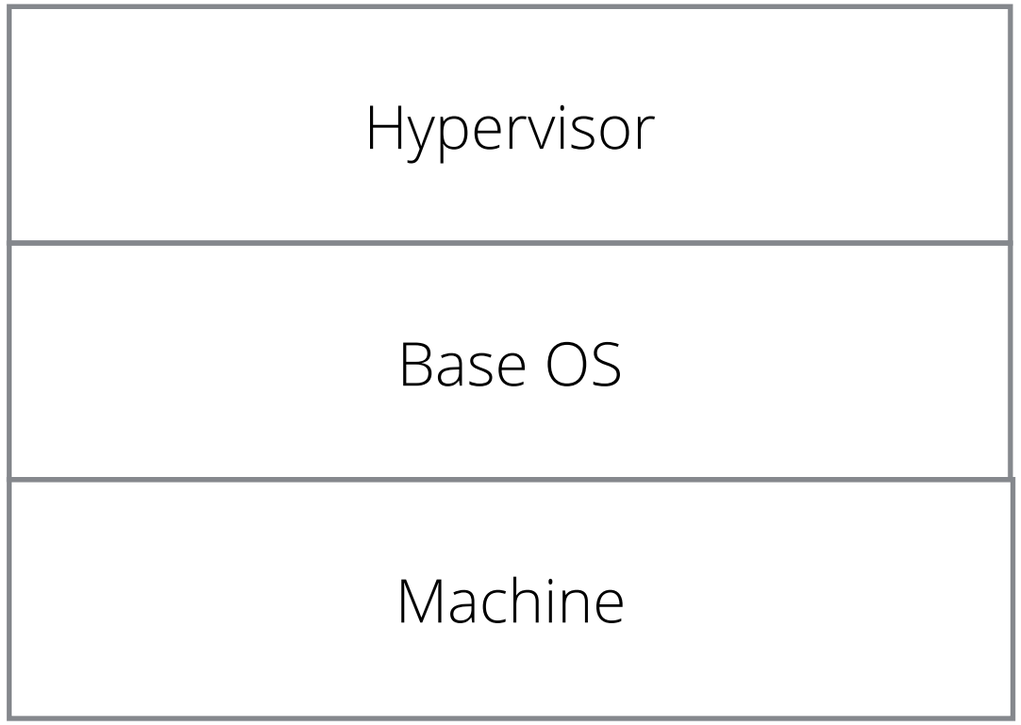
STANDARD VIRTUALISATION

STANDARD VIRTUALISATION

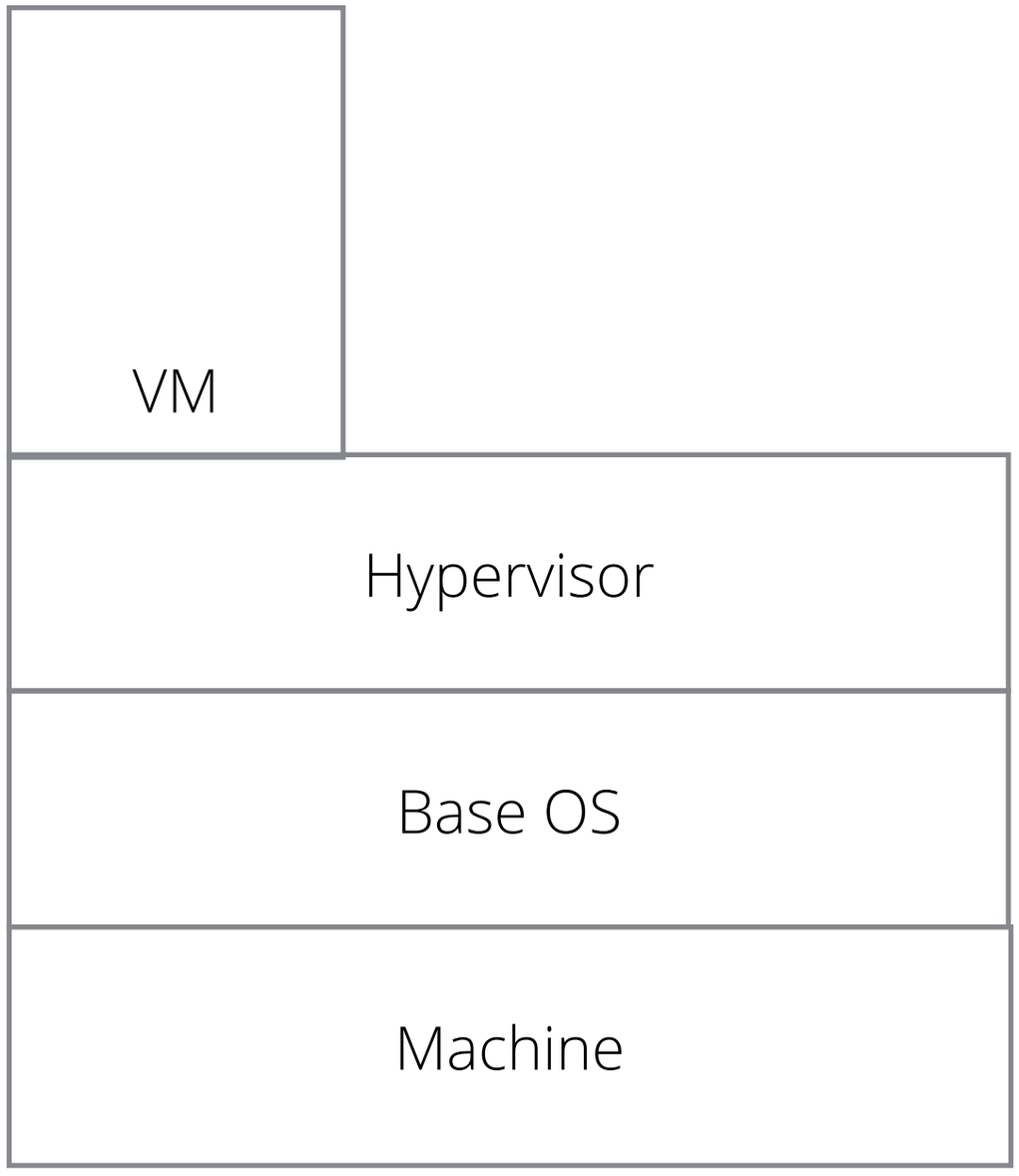
STANDARD VIRTUALISATION



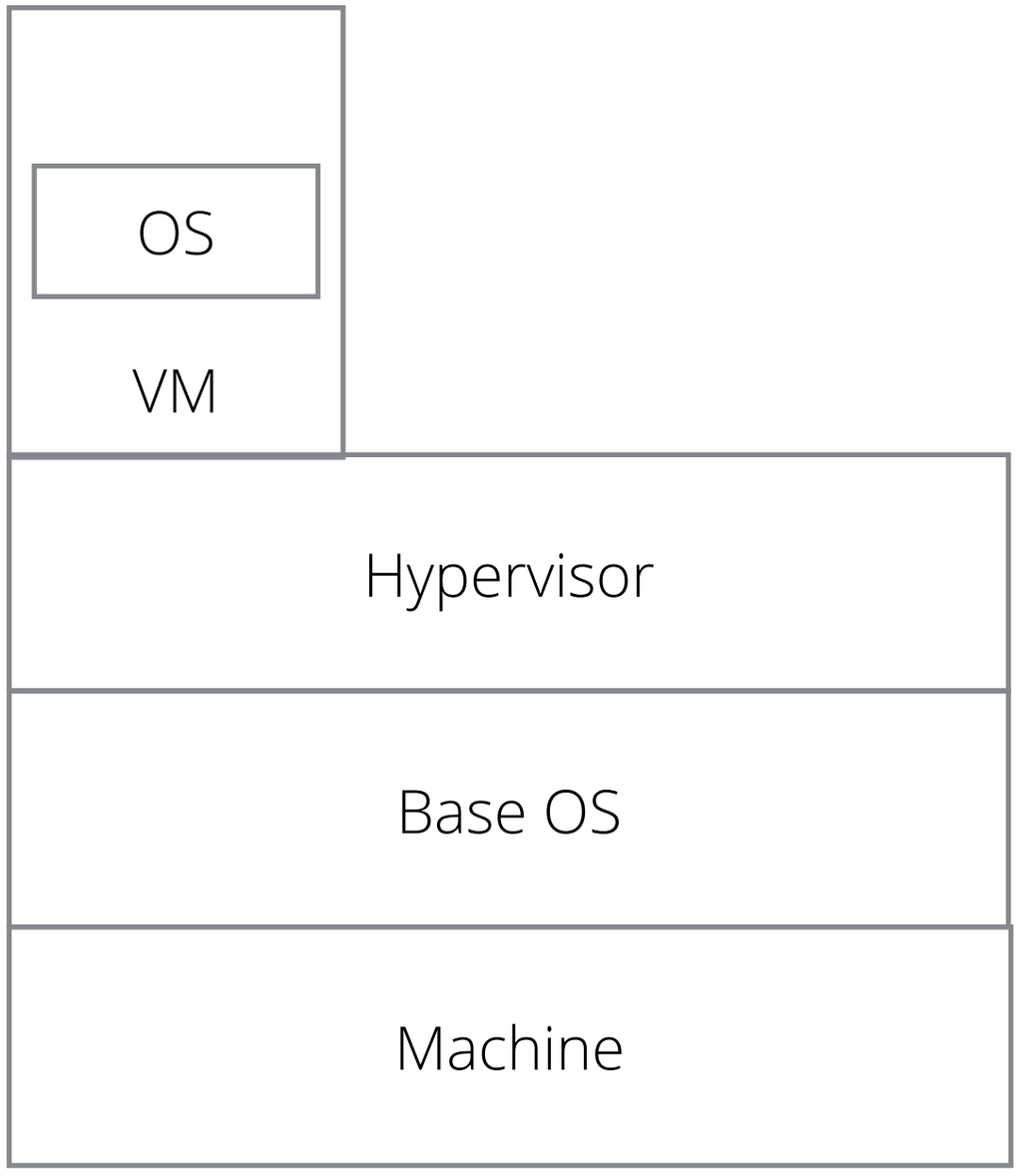
STANDARD VIRTUALISATION



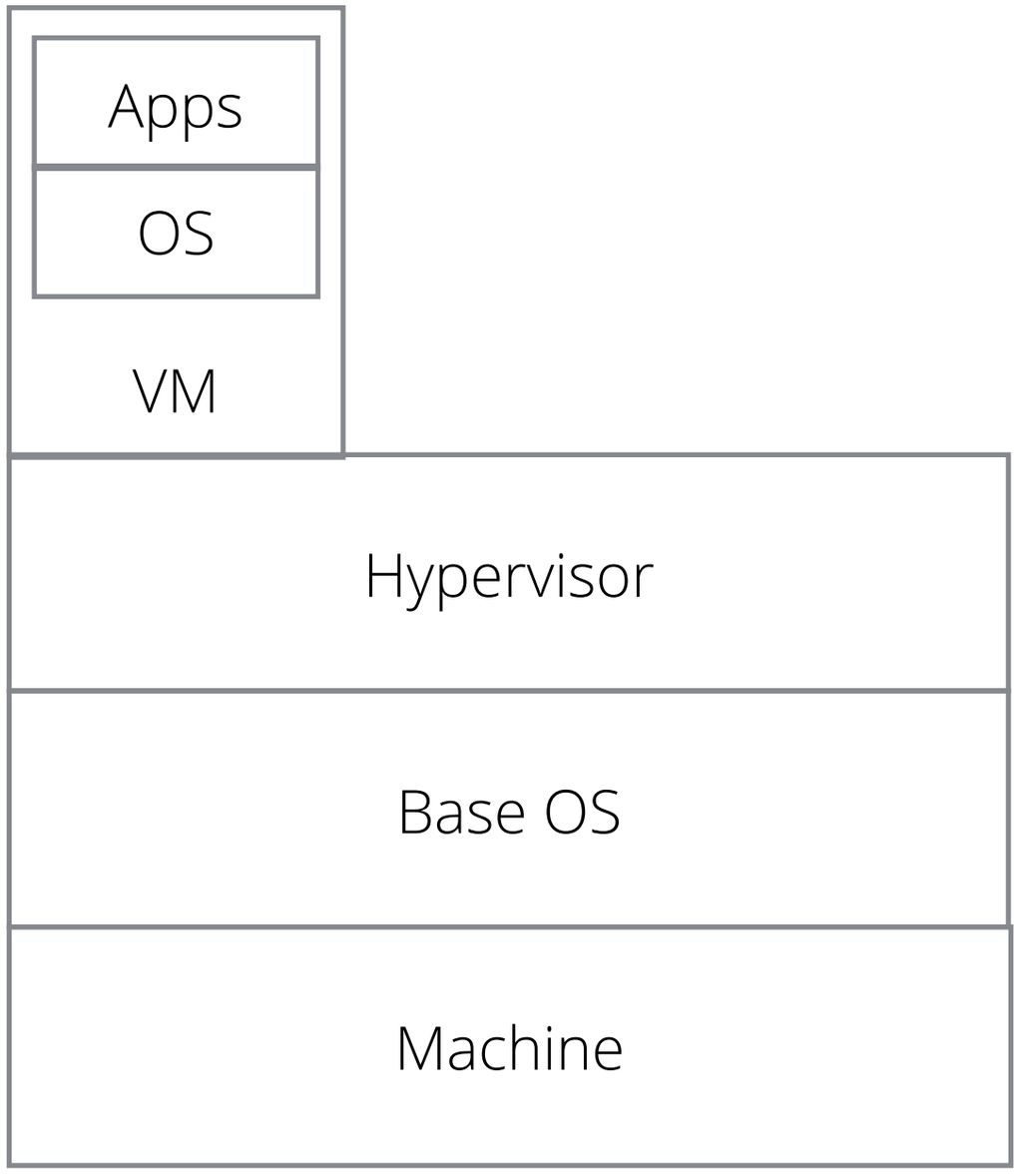
STANDARD VIRTUALISATION



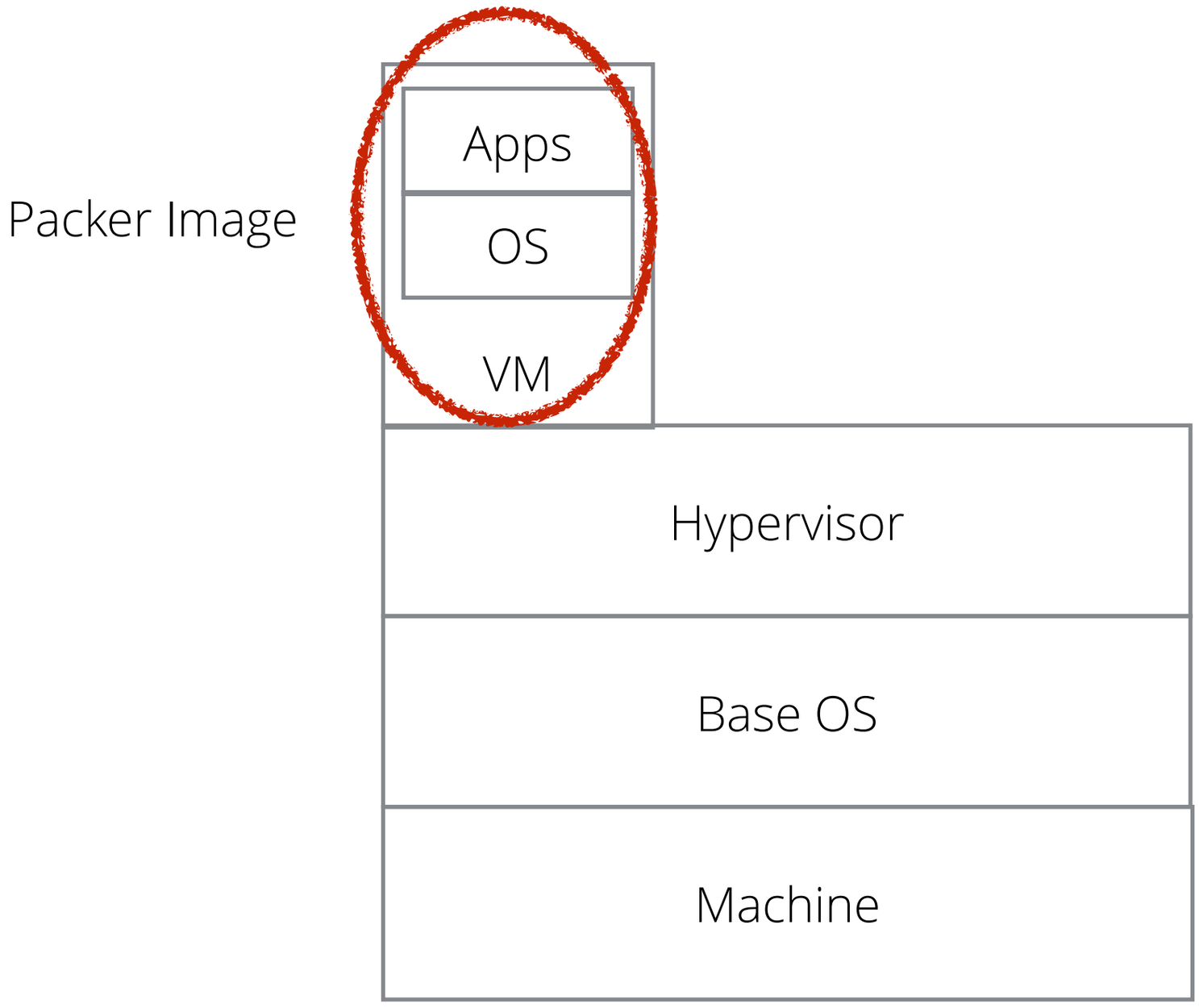
STANDARD VIRTUALISATION



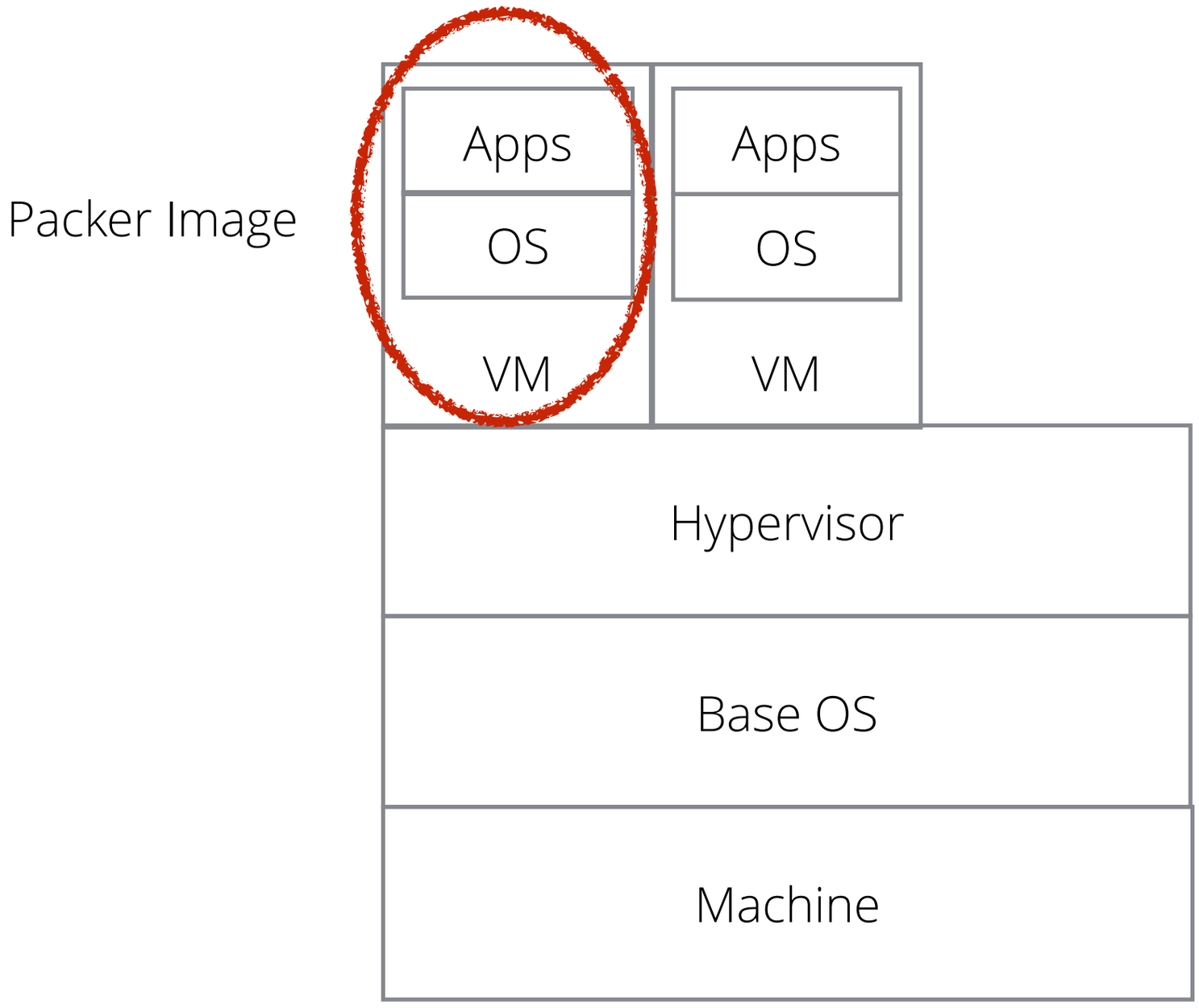
STANDARD VIRTUALISATION



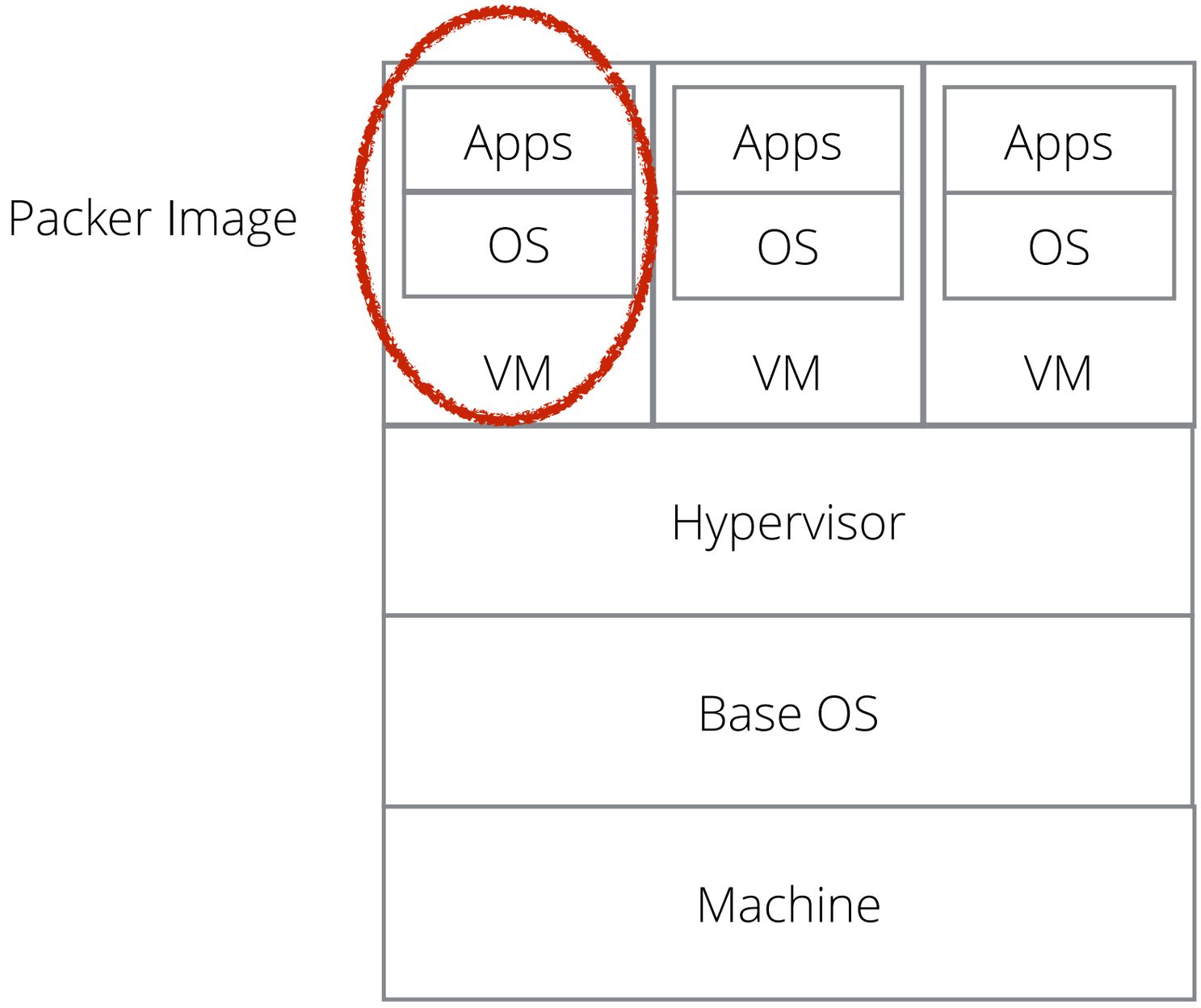
STANDARD VIRTUALISATION



STANDARD VIRTUALISATION



STANDARD VIRTUALISATION



LXC – Linux Containers

Userspace tools for the Linux kernel containers

Home News Downloads Mailing-lists

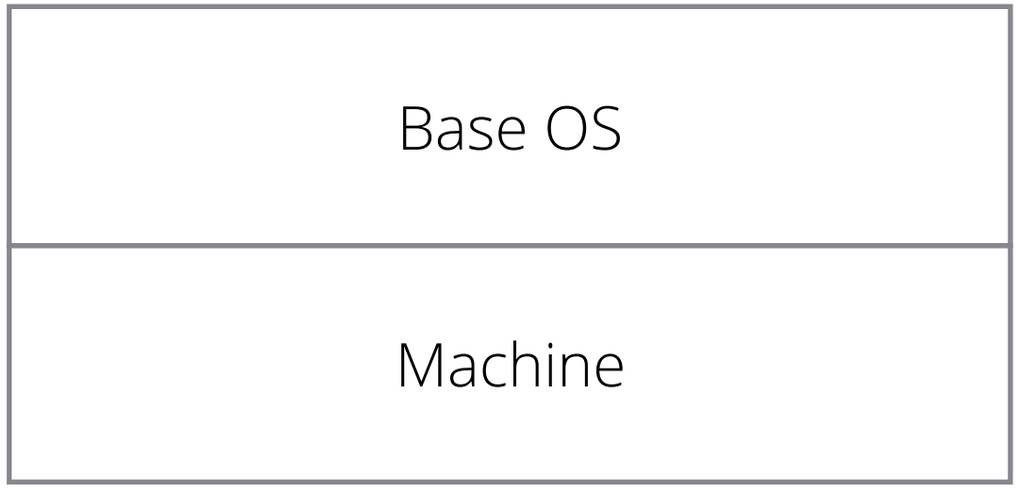
What's LXC?

LXC is a userspace interface for the Linux kernel containment features.
Through a powerful API and simple tools, it lets Linux users easily create and manage system or application containers.

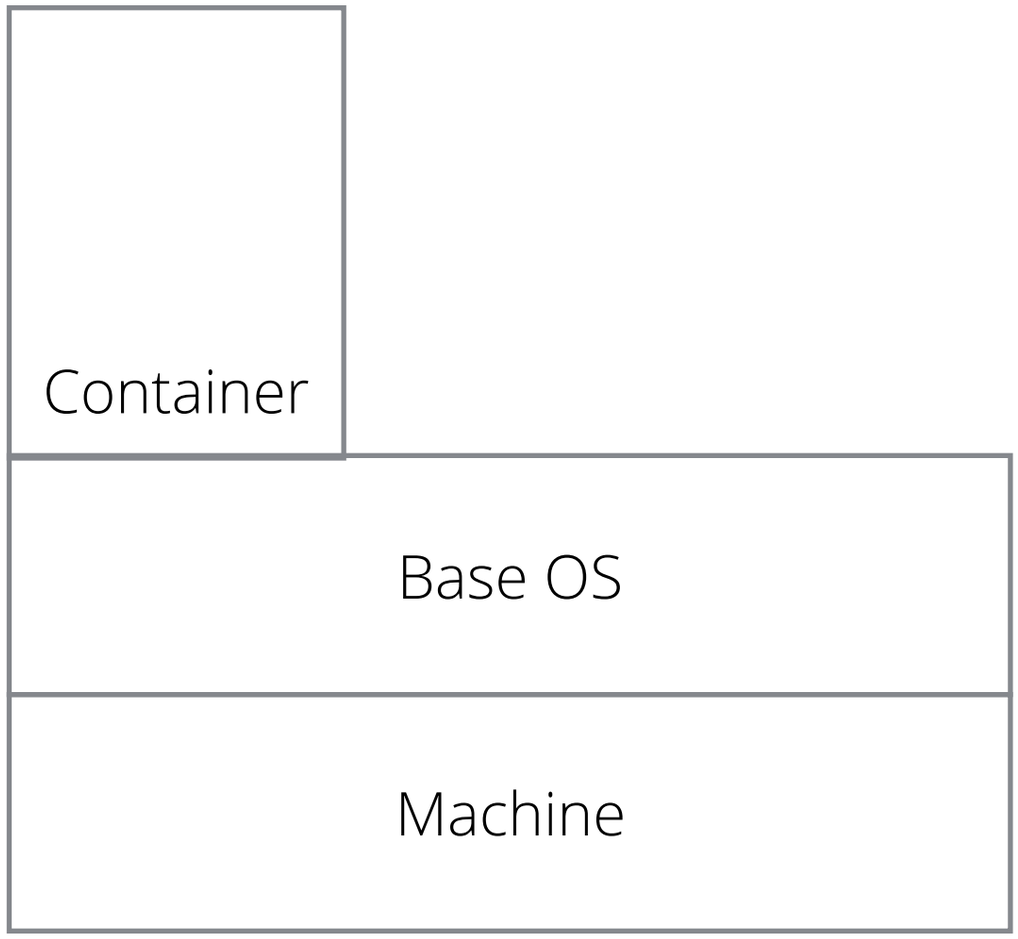
CONTAINER VIRTUALISATION

CONTAINER VIRTUALISATION

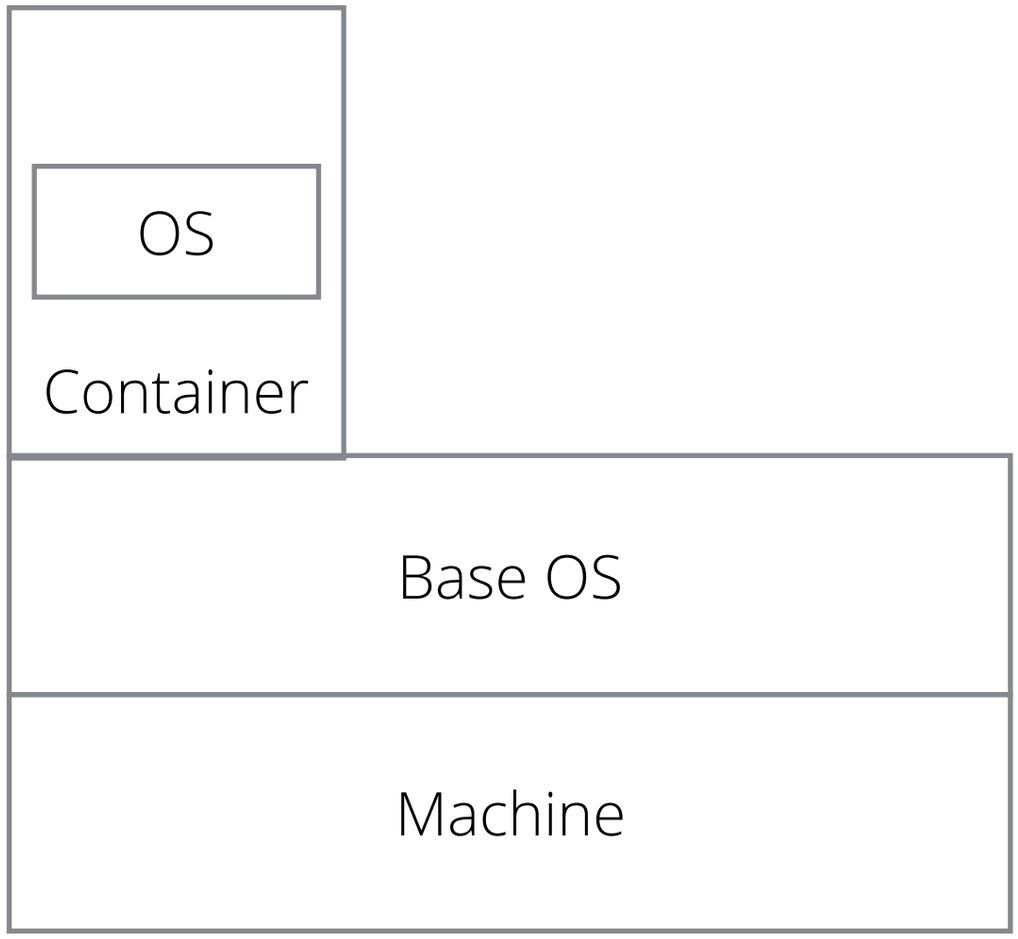
CONTAINER VIRTUALISATION



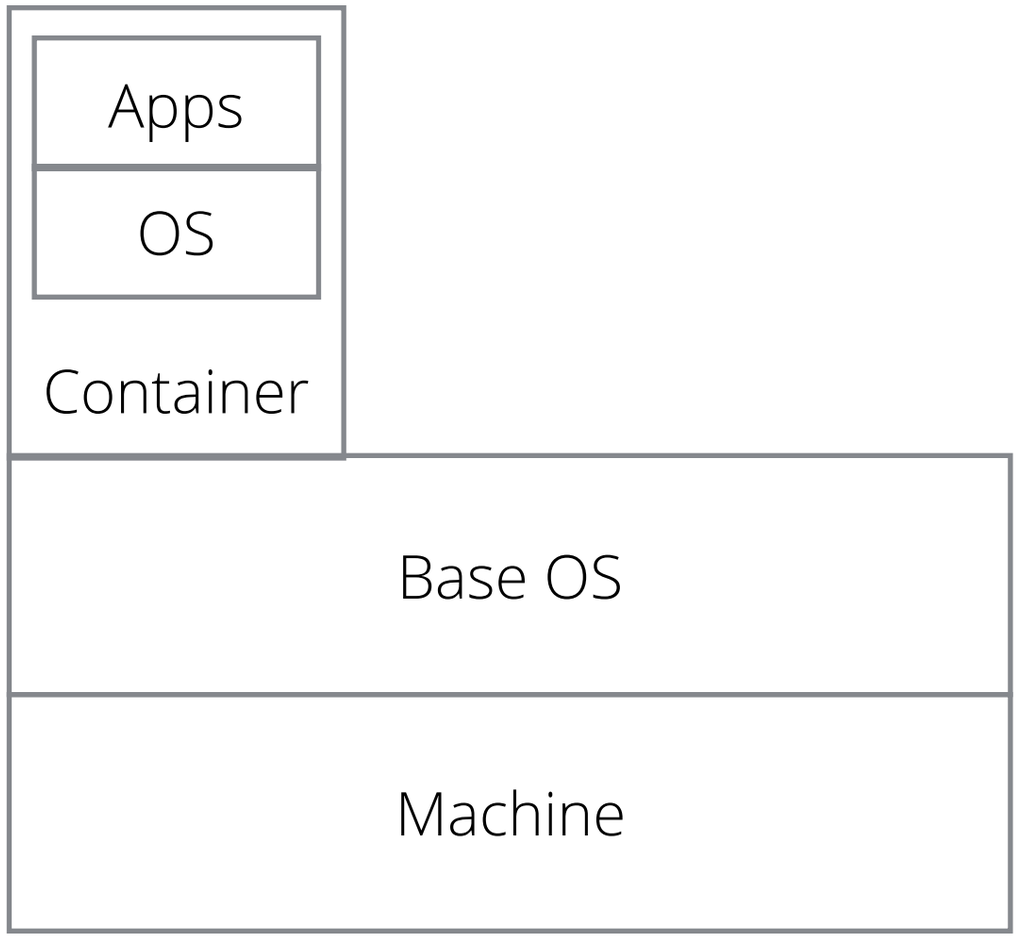
CONTAINER VIRTUALISATION



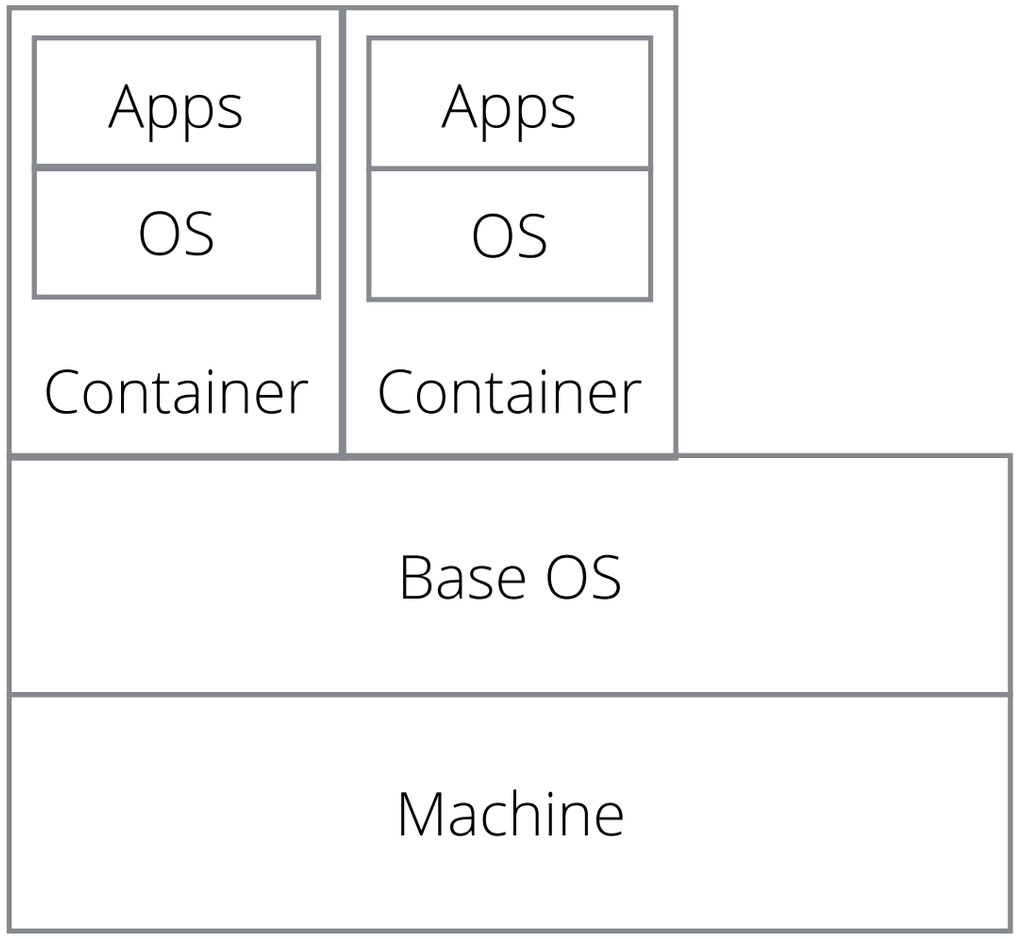
CONTAINER VIRTUALISATION



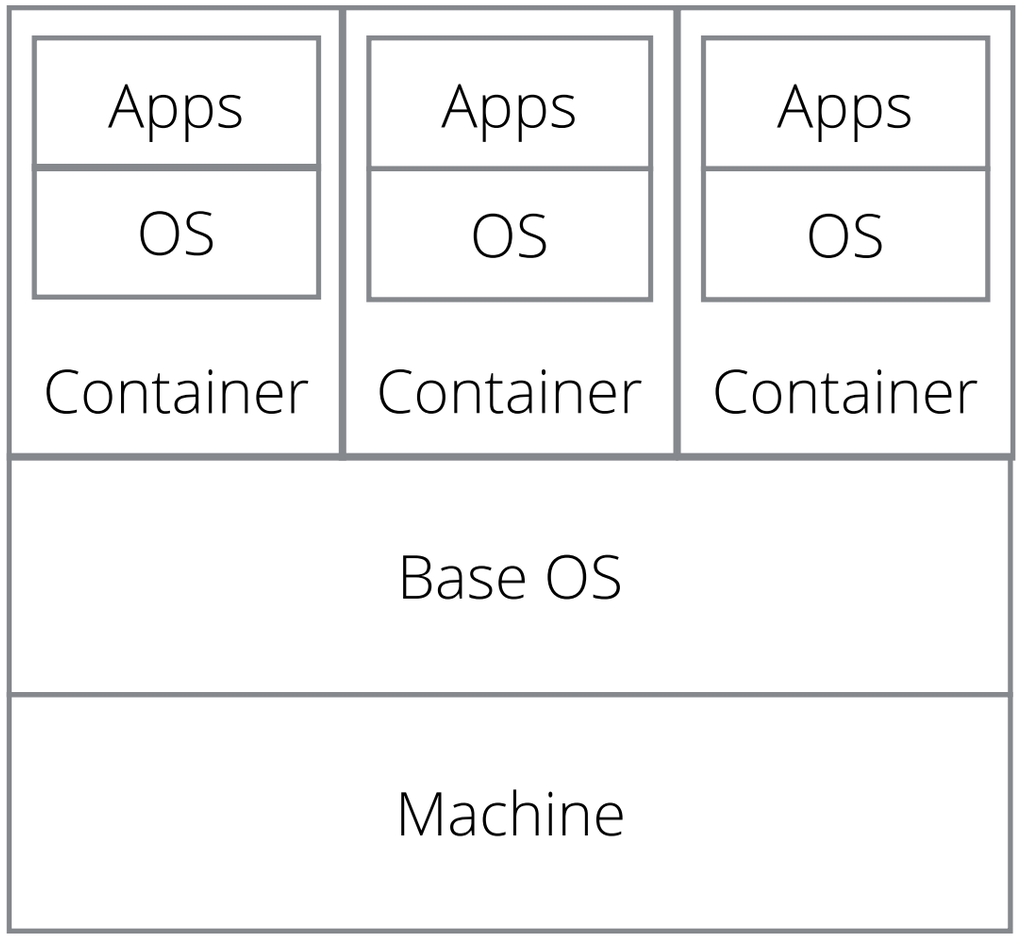
CONTAINER VIRTUALISATION



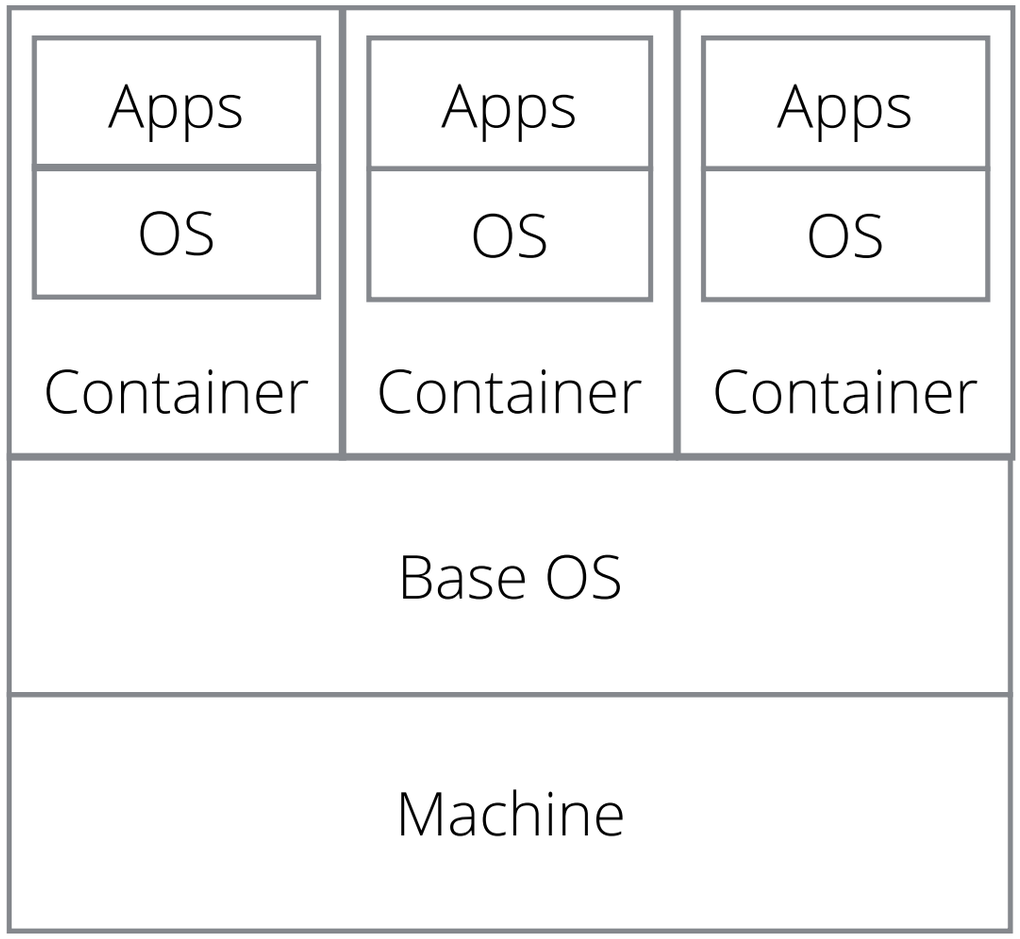
CONTAINER VIRTUALISATION



CONTAINER VIRTUALISATION

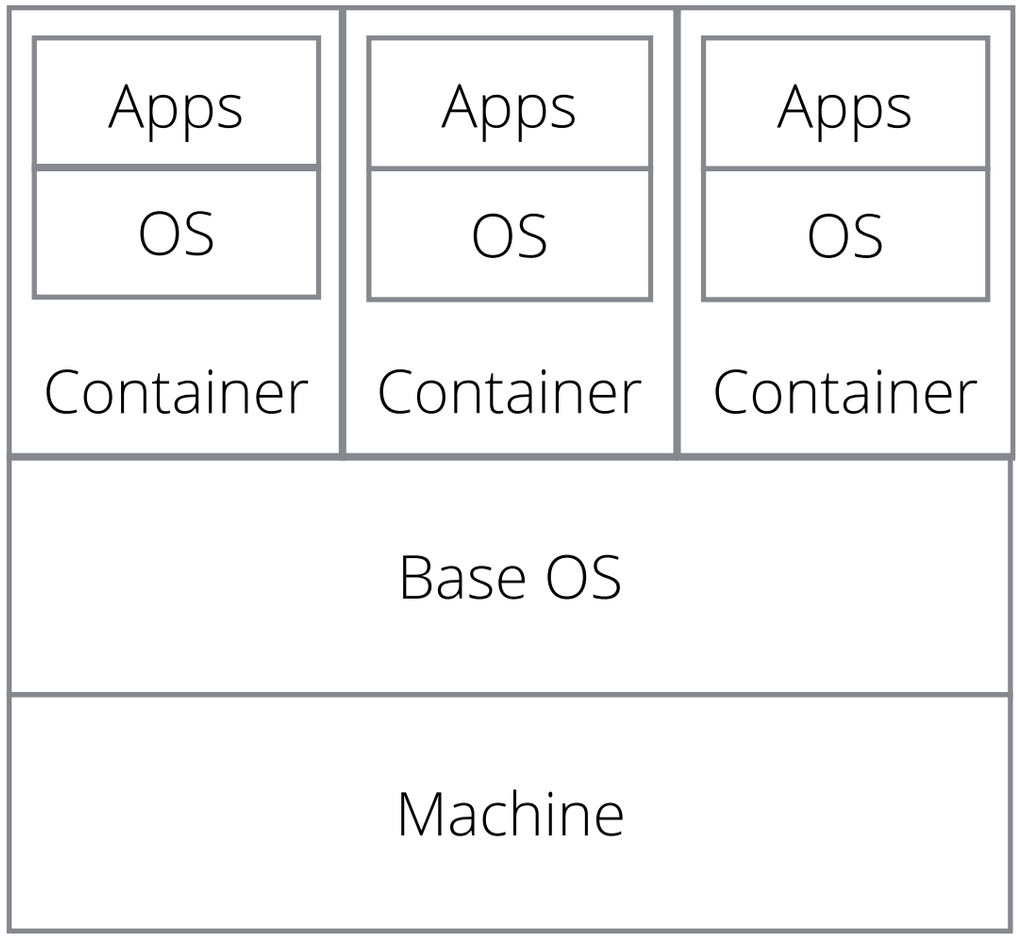


CONTAINER VIRTUALISATION



Linux Only

CONTAINER VIRTUALISATION

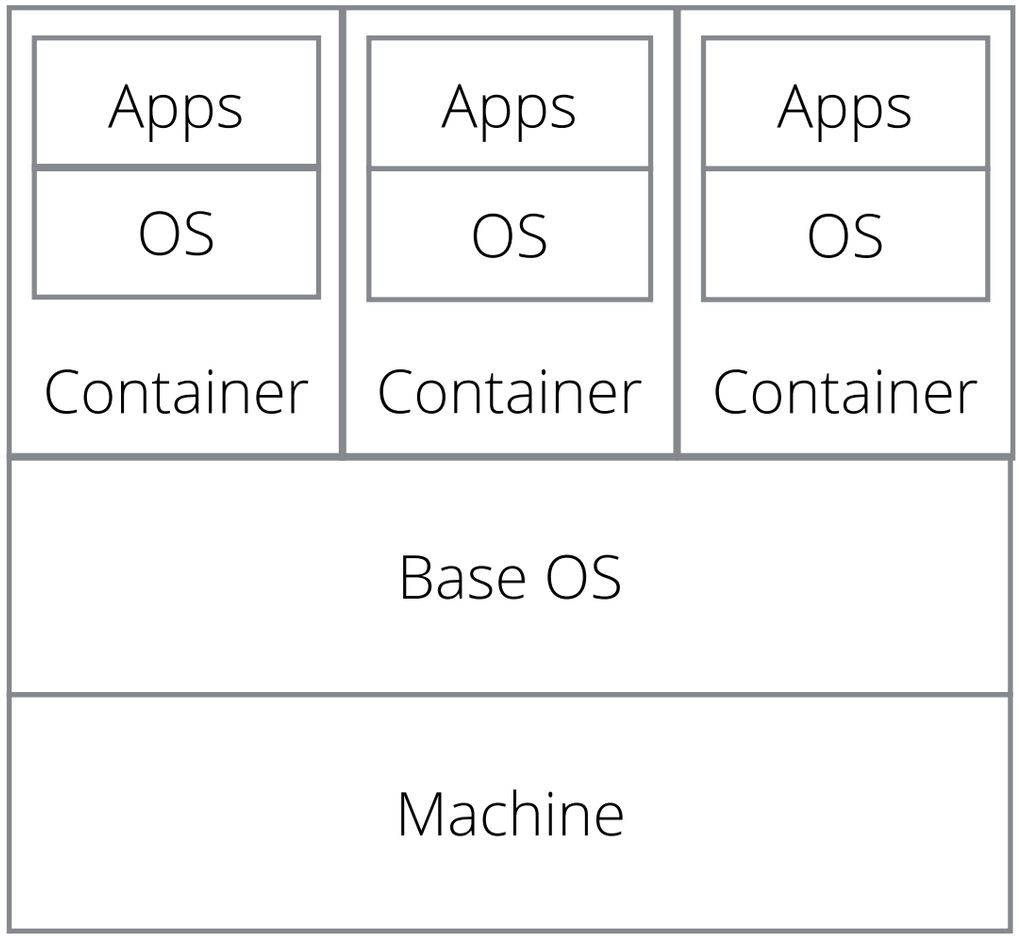


Linux Only

Same Kernel

CONTAINER VIRTUALISATION

Fine-grained control



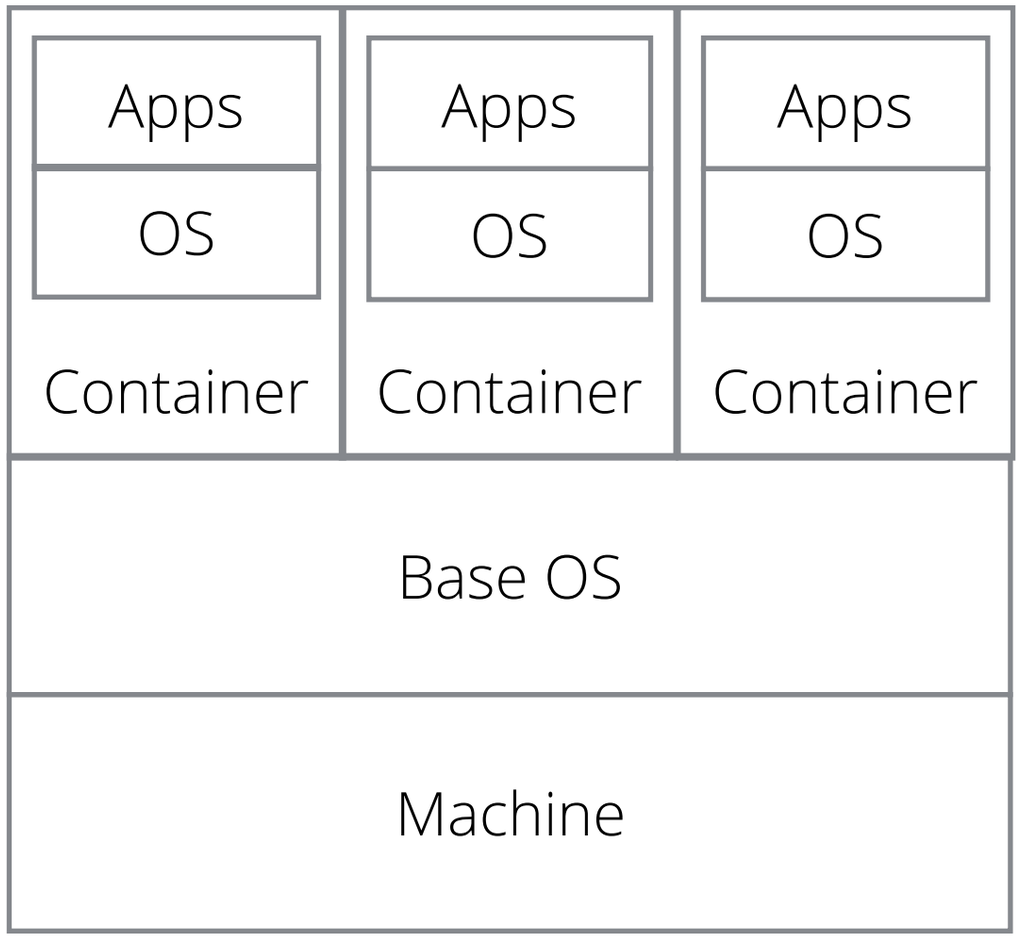
Linux Only

Same Kernel

CONTAINER VIRTUALISATION

Fine-grained control

Very fast to provision



Linux Only

Same Kernel

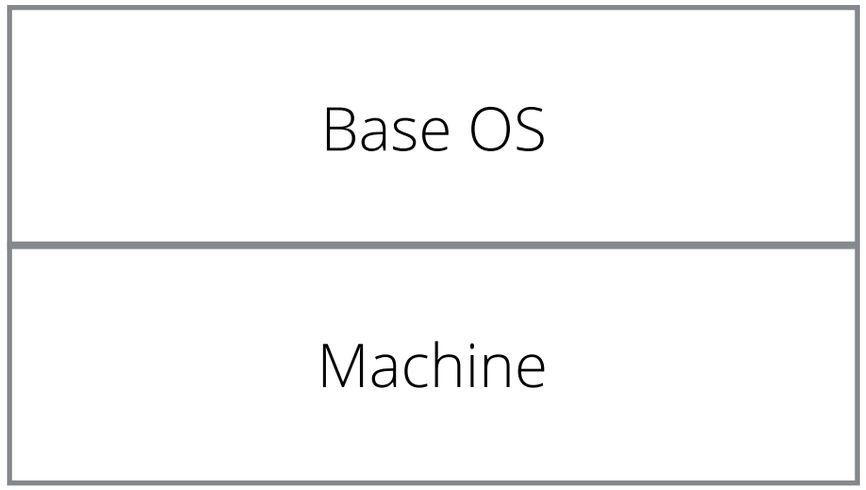
#geecon

@samnewman

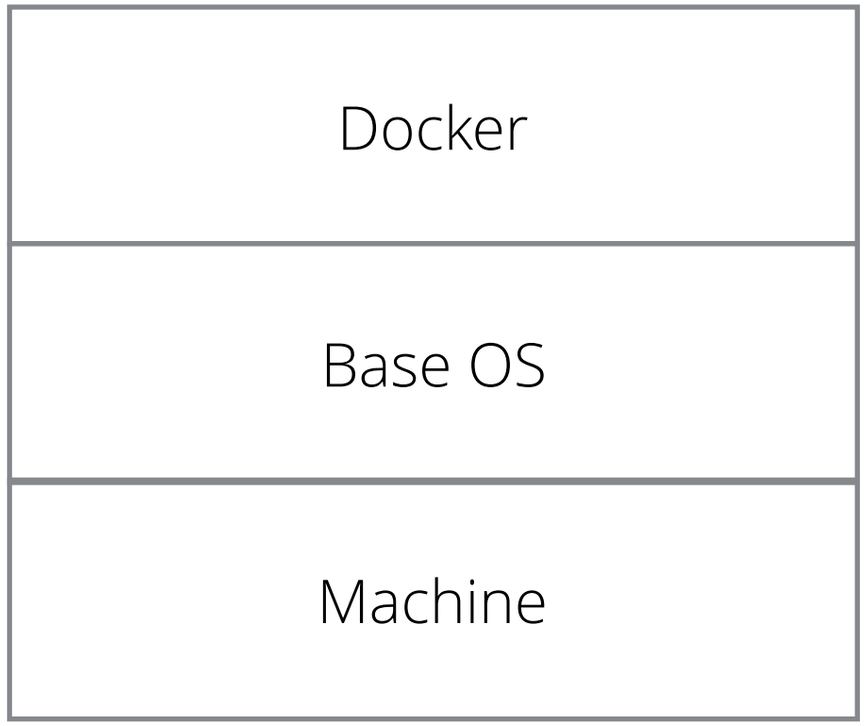
DOCKER

DOCKER

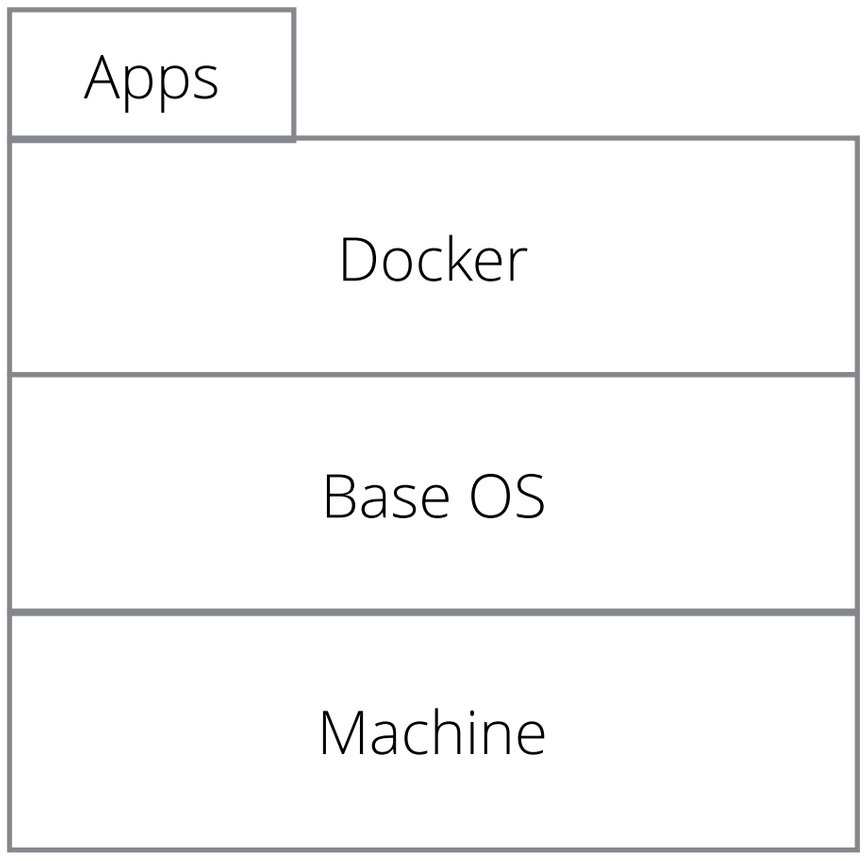
DOCKER



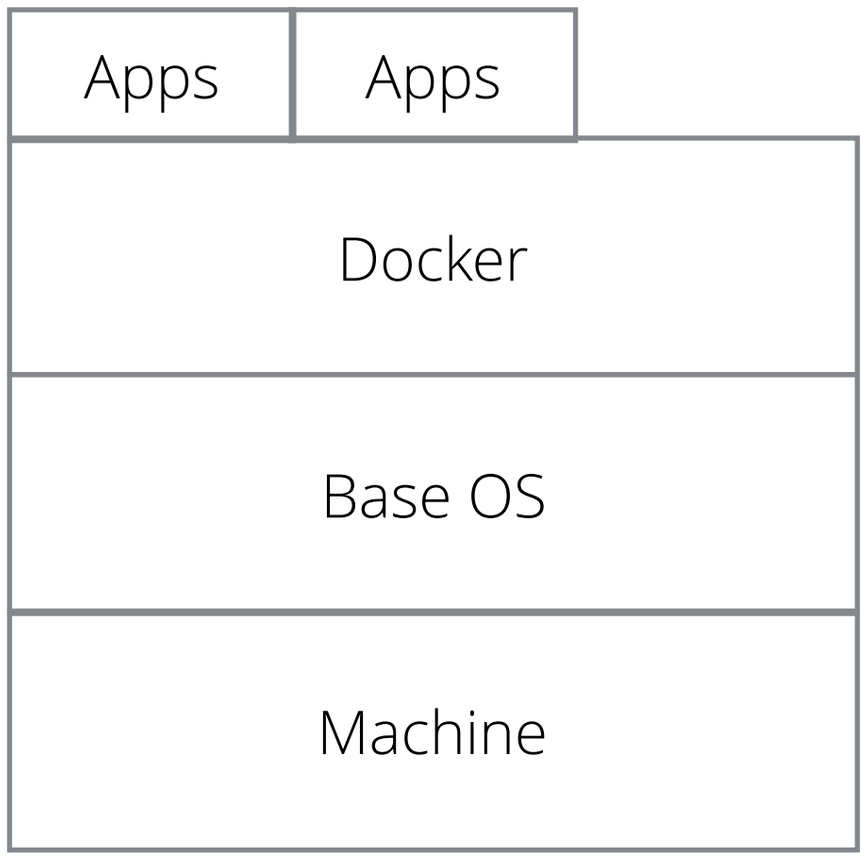
DOCKER



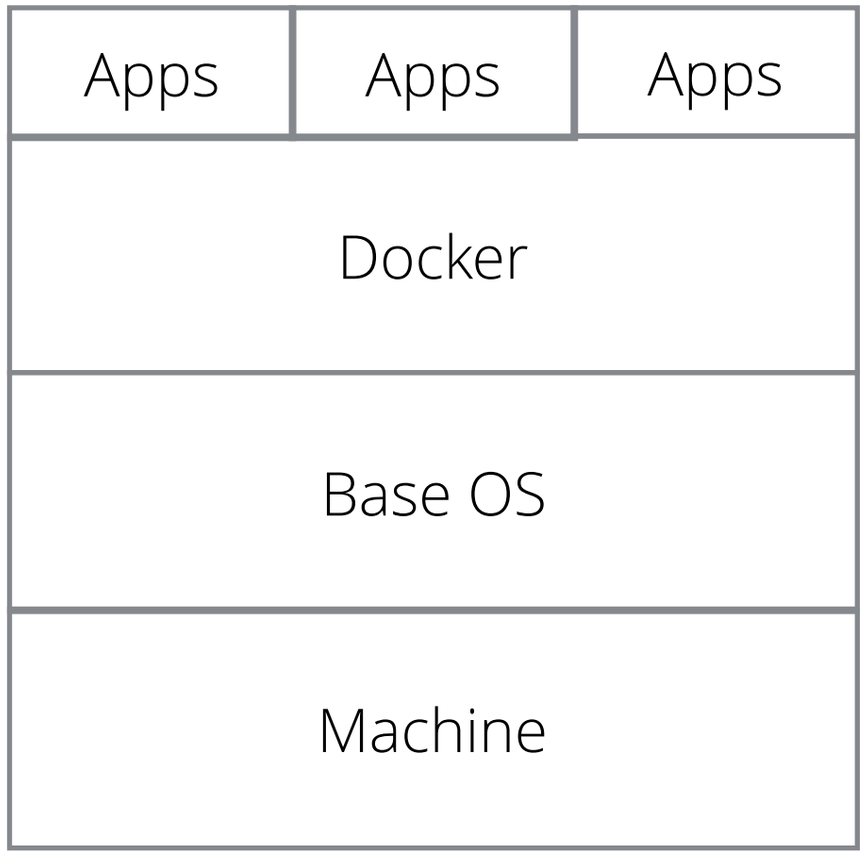
DOCKER



DOCKER

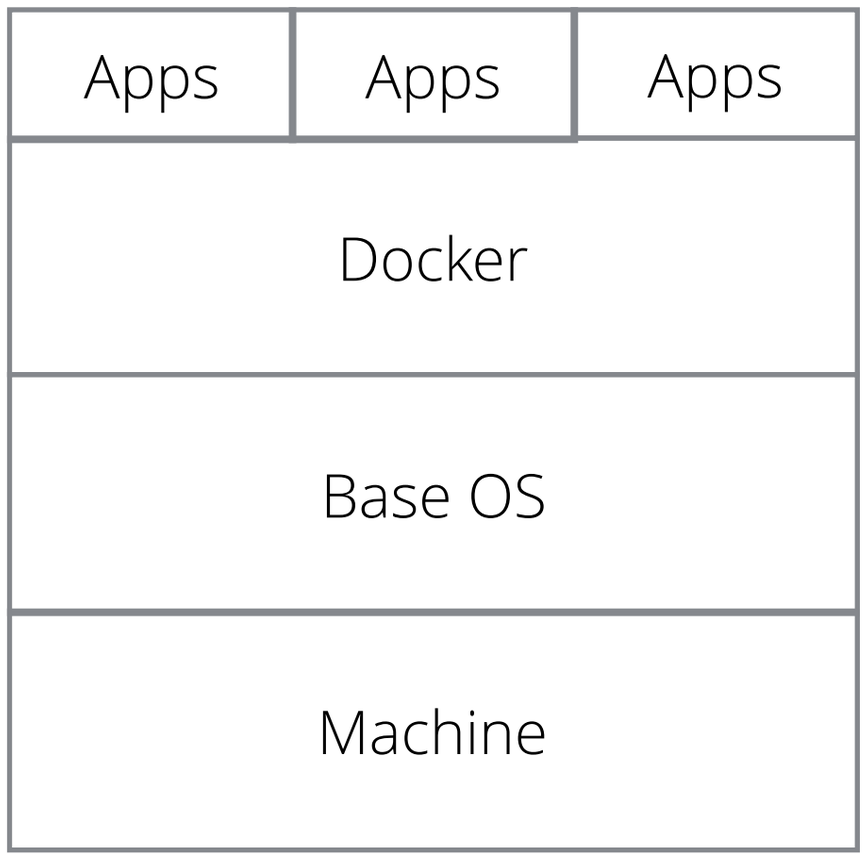


DOCKER

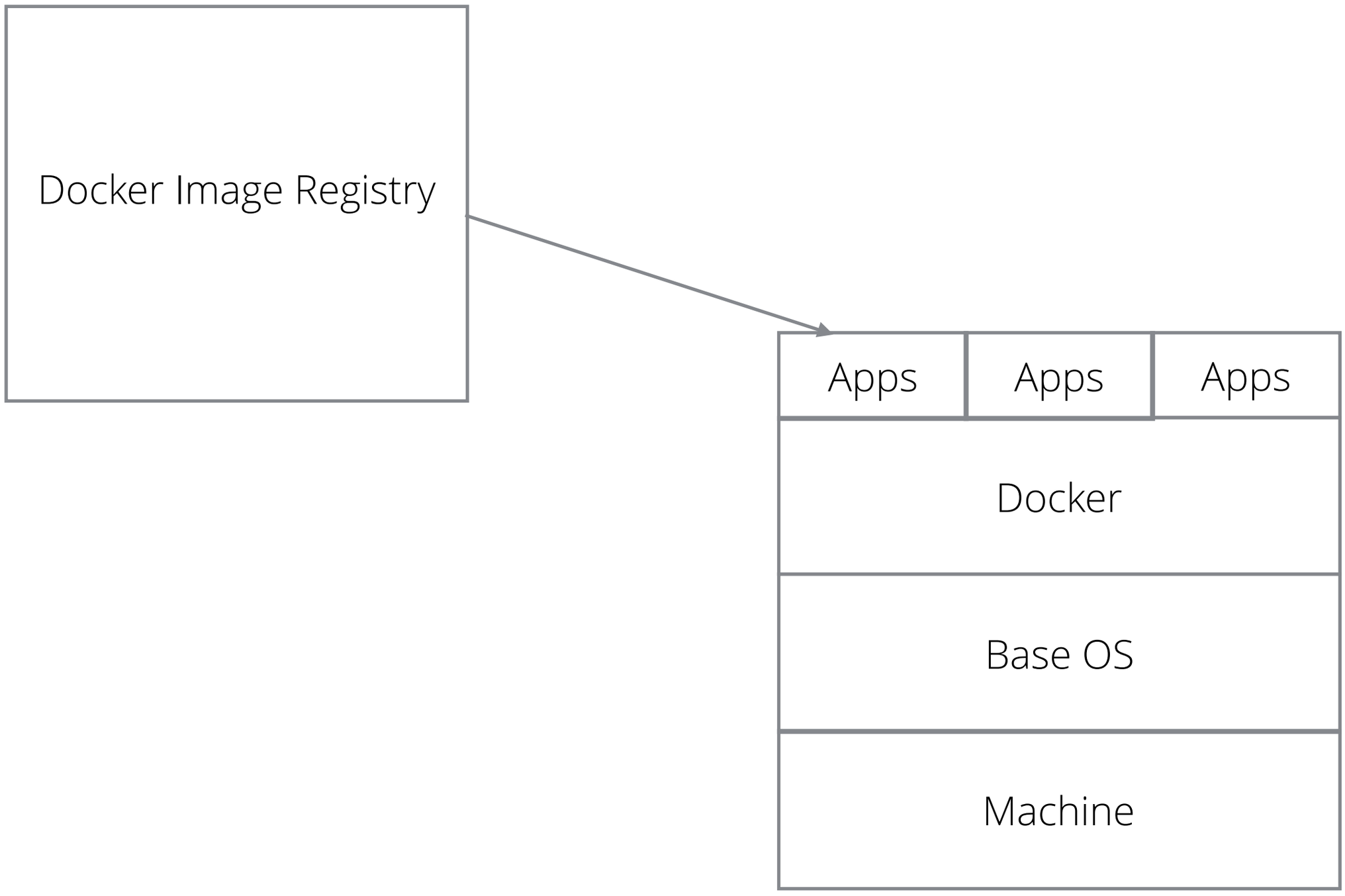


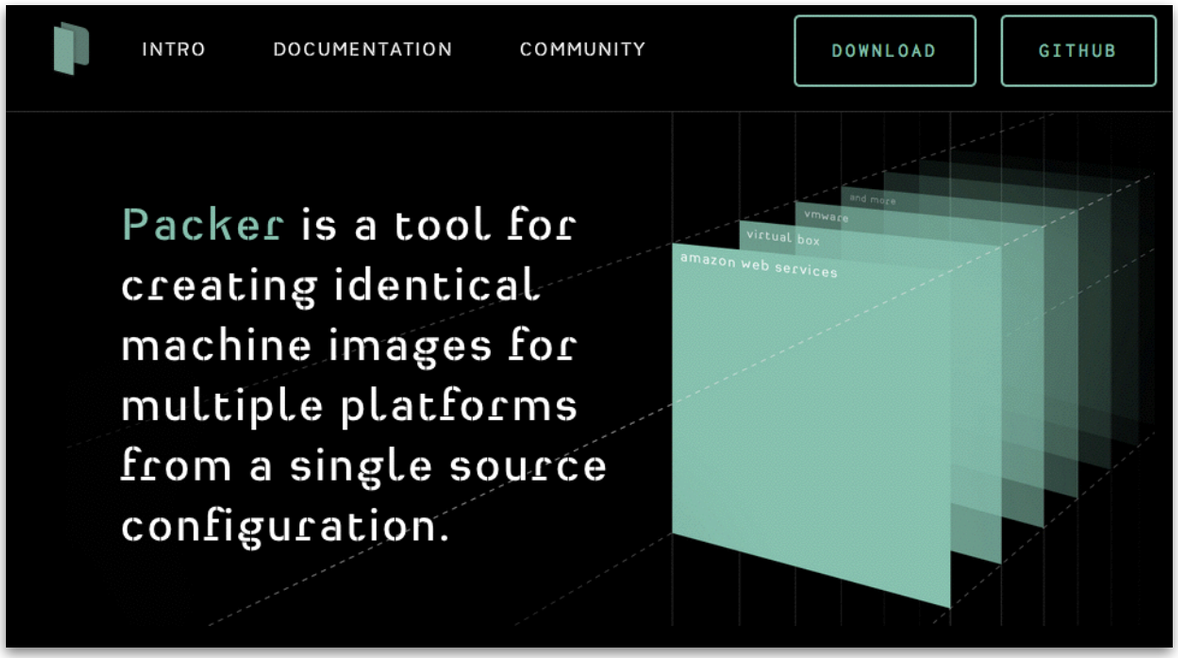
DOCKER

Docker Image Registry



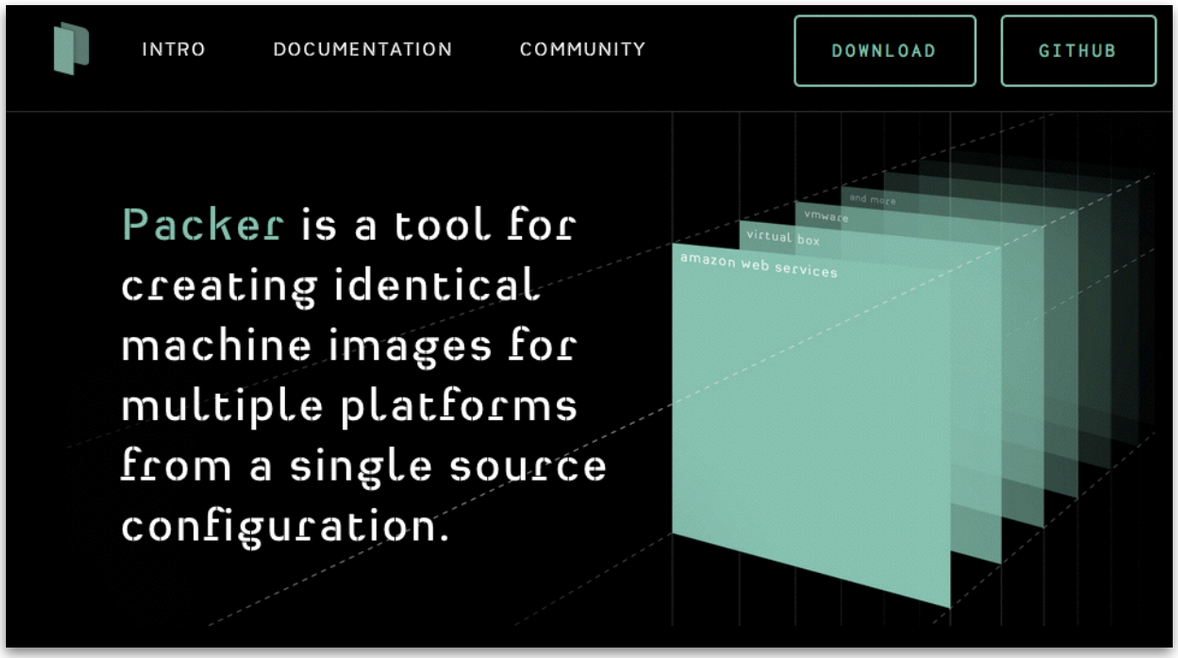
DOCKER


Docker Image Registry
Apps
Apps
Apps
Docker
Base OS
Machine



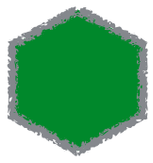
#geecon

@samnewman

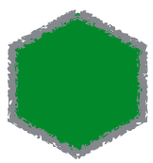

INTRO
DOCUMENTATION
COMMUNITY
DOWNLOAD
GITHUB
Packer is a tool for creating identical machine images for multiple platforms from a single source configuration.
and more
vmware
virtual box
amazon web services


docker
Home
Learn More
Getting started
Community
Documentation
Blog
INDEX
sign up
an open source project to pack, ship and run any application as a lightweight container
docker


Build
S/M Tests
Large Tests
UAT
Prod



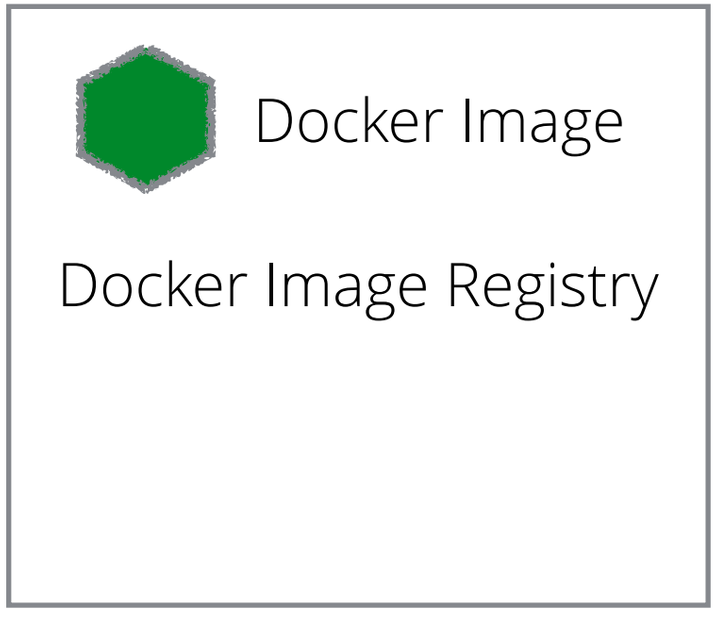
Docker Image

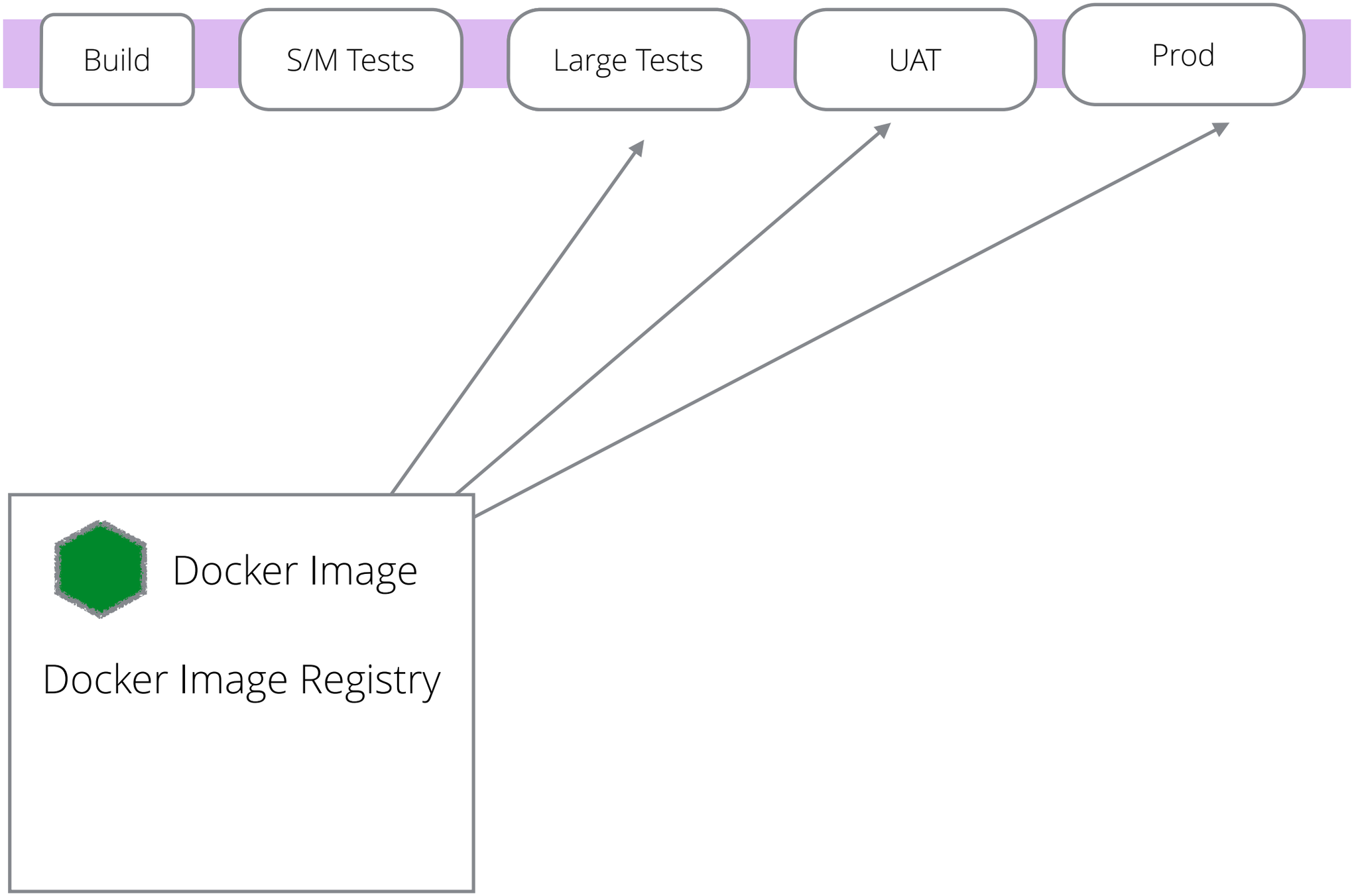


Docker Image

Docker Image Registry


Build
S/M Tests
Large Tests
UAT
Prod


Docker Image
Docker Image Registry


Build
S/M Tests
Large Tests
UAT
Prod
Docker Image
Docker Image Registry

Be aware of - and balance - your test Pyramid

Be aware of - and balance - your test Pyramid

Understand the balance between testing & rapid remediation

Be aware of - and balance - your test Pyramid

Understand the balance between testing & rapid remediation

Deploy one thing at a time

Be aware of - and balance - your test Pyramid

Understand the balance between testing & rapid remediation

Deploy one thing at a time

Consider consumer-driven contracts over integration tests

Be aware of - and balance - your test Pyramid

Understand the balance between testing & rapid remediation

Deploy one thing at a time

Consider consumer-driven contracts over integration tests

Explore image-based deployments to reduce environment differences

THANKS!

Any questions:
@samnewman
snewman@thoughtworks.com

ThoughtWorks®