A BLUEPRINT FOR

SCALA MICROSERVICES

FEDERICO FEROLDI
CTO / MEASURENCE.COM
@CLOUDIFY

WHO IS THIS GUY?

Technologies	
20 years ago	C / C++ / x86 ASM
15 years ago	C / C++ / Perl / PHP
10 years ago	Ruby / Perl / JS / Java
last 5 years	Scala / Ruby / JS / LUA
Worked at	Y! Jumpy MOBANGO m Gild GILT
Founded	Cloudify Coderloop measurence

OUR GOAL

Empower developers with tools and process to own a service from development to production.

RATIONALE

Small team: time is scarce

Heavy use of open source

Automate al the things

Sane standards, easy override

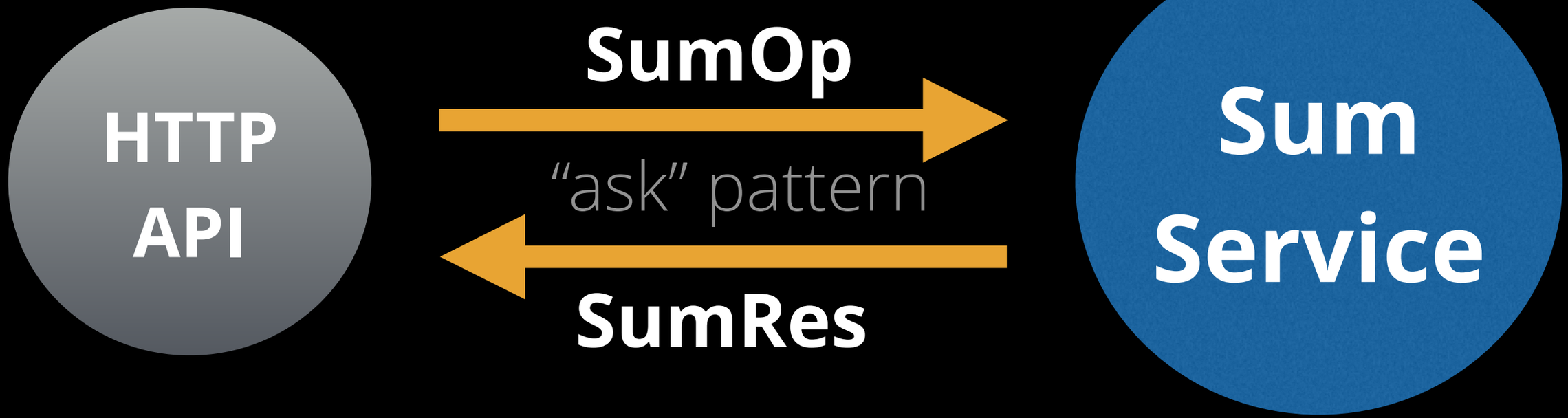
MAIN TOPICS

- Minimum Viable Service
- Deployment & Discovery
- Best practices

MINIMUM VIABLE SERVICE

- Business Logic
- REST API
- Configuration
- Health / Monitoring
- Releasing / Versioning
- Packaging

BUSINESS LOGIC = AKKA ACTORS



BUSINESS LOGIC = AKKA ACTORS

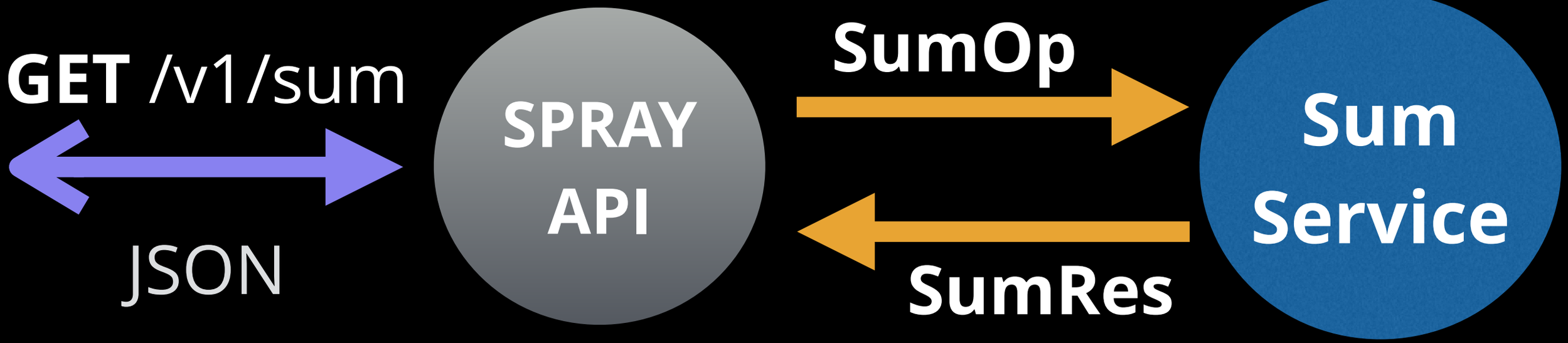
```
object SumService {
  def props = Props(classOf[SumService])

  case class SumOp(a: Int, b: Int)
  case class SumRes(result: Int)
}
```

```
class SumService extends Actor {
  import SumService._

  def receive: Receive = {
    case SumOp(a, b) => sender() ! SumRes(a + b)
  }
}
```

REST API = SPRAY.IO



REST API = SPRAY.IO

```
case class SumApiResponse(result: Int)

object SumProtocol extends DefaultJsonProtocol {
  implicit val sumApiResponseFormat = jsonFormat1(SumApiResponse)
}

class SumHttpServiceV1(services: Services) {
  import SumProtocol._

  def route = path("v1" / "sum") {
    get {
      parameters('a.as[Int], 'b.as[Int]) { (a, b) =>
        def future = (services.sumService ? SumOp(a, b)).mapTo[SumRes]
        onSuccess(future) { response =>
          complete(SumApiResponse(response.result))
        }
      }
    }
  }
}
```

API DOCS = SWAGGER

GET /

HTML/JSON docs

SPRAY-SWAGGER

API DOCS = SWAGGER

```
@Api(value = “/v1/sum", description = "Sum numbers.")
class SumHttpServiceV1(services: Services) {
  ...
```

API DOCS = SWAGGER

```
@ApiOperation(value = "Returns the sum”, httpMethod = “GET")

@ApiImplicitParams(Array(
  new ApiImplicitParam(
    name="a", required=true, dataType="integer", paramType=“query"
  ),
  new ApiImplicitParam(
    name="b", required=true, dataType="integer", paramType=“query"
  )
))

@ApiResponses(Array(
  new ApiResponse(
    code=200, message="The sum", response=classOf[SumApiResponse]
  )
))

def route = path("v1" / "sum") { … }
```

API DOCS = SWAGGER

```
@ApiModel(description = "Result of the sum")
case class SumApiResponse(
  @(ApiModelProperty @field)(value = "The sum of a and b")
  result: Int
)
```

REST API = SPRAY.IO + SWAGGER

sum : Operations on numbers.

Show/Hide | List Operations | Expand Operations | Raw

GET /v1/sum — Returns the sum of two numbers

Parameters

Parameter	Value	Description	Parameter Type	Data Type
a	(required)	**first number**	query	integer
b	(required)	**second number**	query	integer

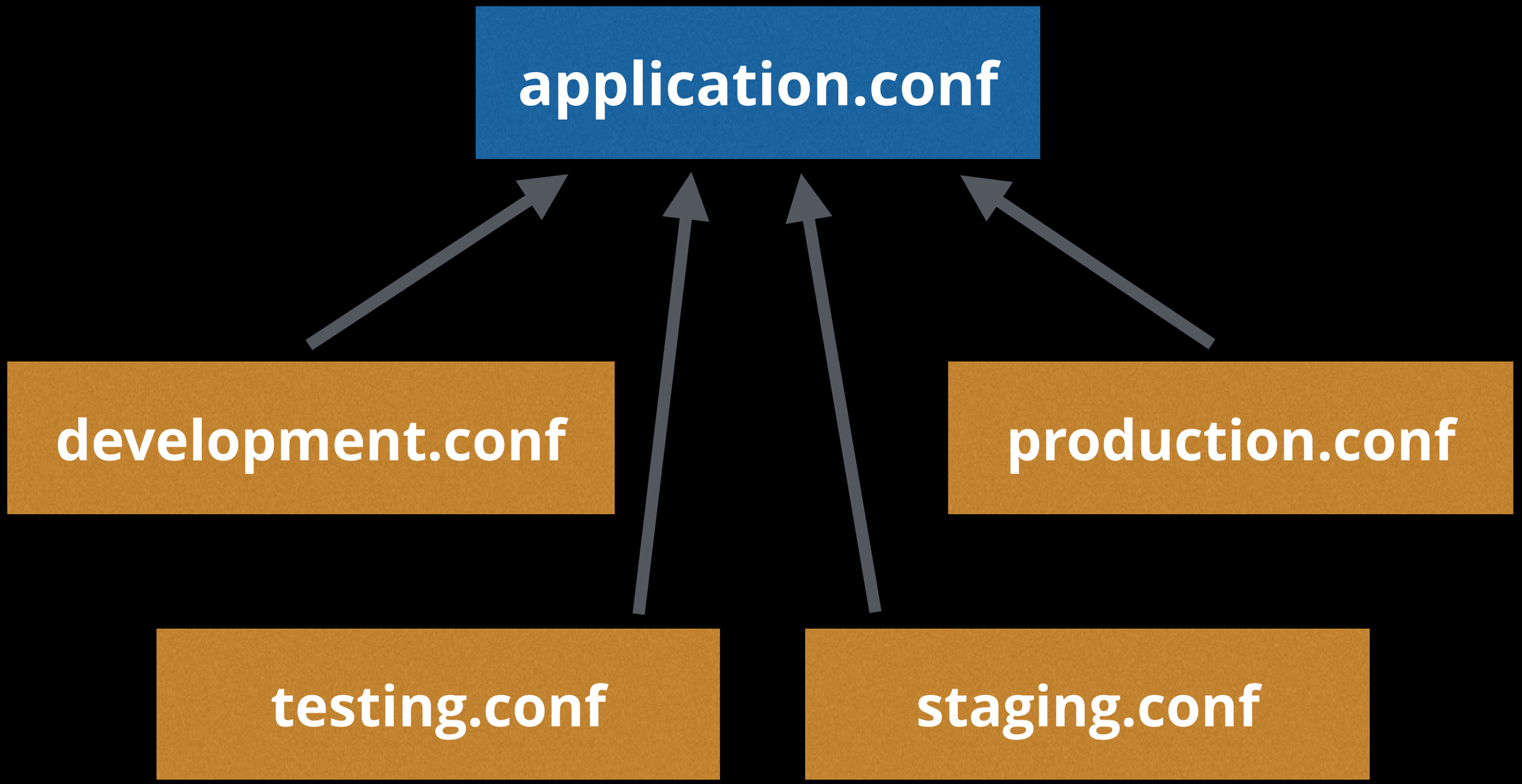
Response Messages

HTTP Status Code	Reason	Response Model
200	The sum of two numbers	Model | Model Schema **SumApiResponse {** **result** (integer, *optional*): The result of the sum **}**

Try it out!

[BASE URL: http://localhost:9999/development/svc-calculator/api-docs , API VERSION: 1.0]

CONFIG = TYPESAFE CONFIG



CONFIG = TYPESAFE CONFIG

```
akka.loglevel = "INFO"

metrics.carbon = "graphite.local:2003"

acme {
  environment = "production"

  svc-calculator {
    hostname = “0.0.0.0”
    port = "9999"
  }
}
```

application.conf

CONFIG = TYPESAFE CONFIG

```
include "application"
```

production.conf

CONFIG = TYPESAFE CONFIG

```
include "application"

akka.loglevel = "DEBUG"
metrics.carbon = "localhost:2003"

acme {
  environment = "development"

  svc-calculator {
    hostname = "127.0.0.1"
  }
}
```

development.conf

CONFIG = TYPESAFE CONFIG

```
val config = ConfigFactory.defaultApplication()
```

Loads **application.conf** by default

```
$ sbt -Dconfig.resource=development.conf run
```

Loads **development.conf**

CONFIG = TYPESAFE CONFIG

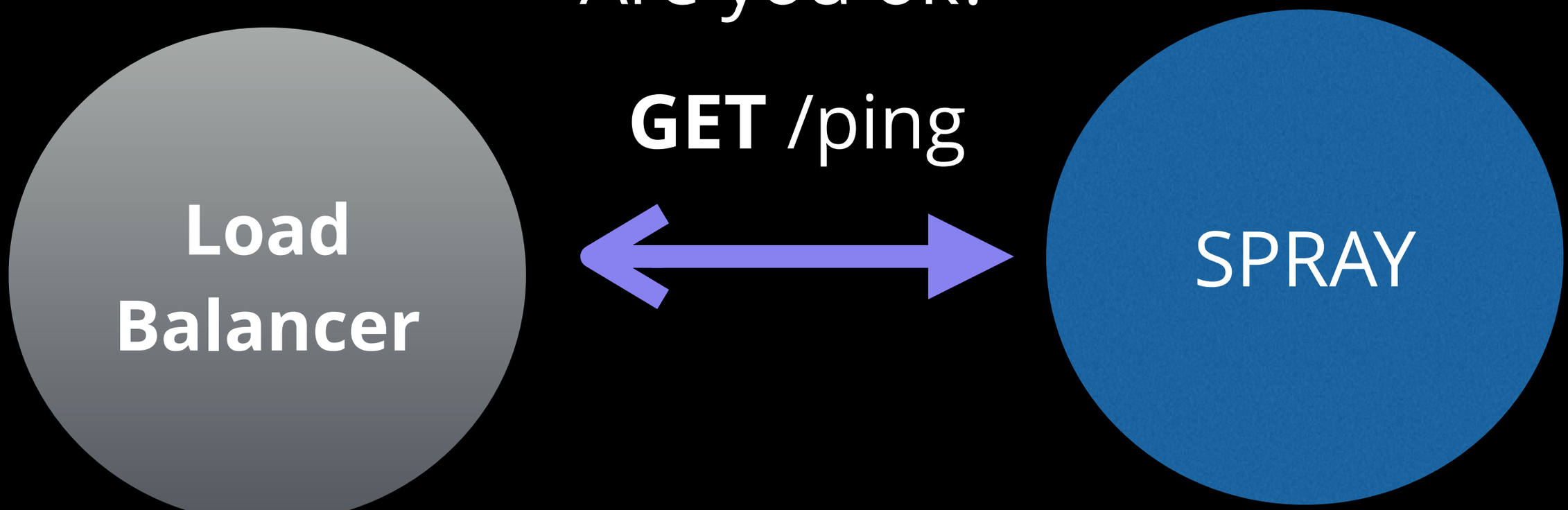
PRO-TIP

Don’t store passwords with source code!

Instead make use ENV vars substitution and keep passwords in a safe place.

```
{
    mysql_username=${MYSQL_USERNAME}
    mysql_password=${MYSQL_PASSWORD}
}
```

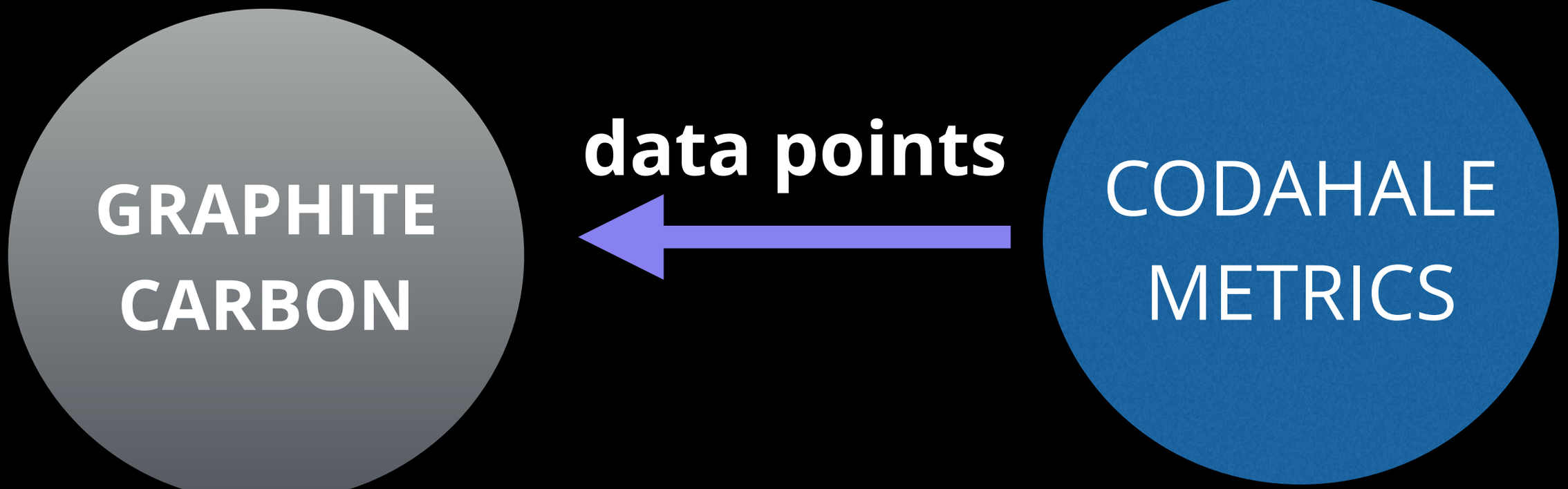
HEALTH = SPRAY + HAPROXY



HEALTH = SPRAY + HAPROXY

```
def route = get {
  path("ping") {
    complete("pong")
  }
}
```

PERFMON = METRICS + GRAPHITE



PERF.MON. = METRICS + GRAPHITE

```scala
object MetricsInstrumentation {
  val metricRegistry = new MetricRegistry()
}

trait Instrumented extends InstrumentedBuilder {
  val metricRegistry = MetricsInstrumentation.metricRegistry
}
```

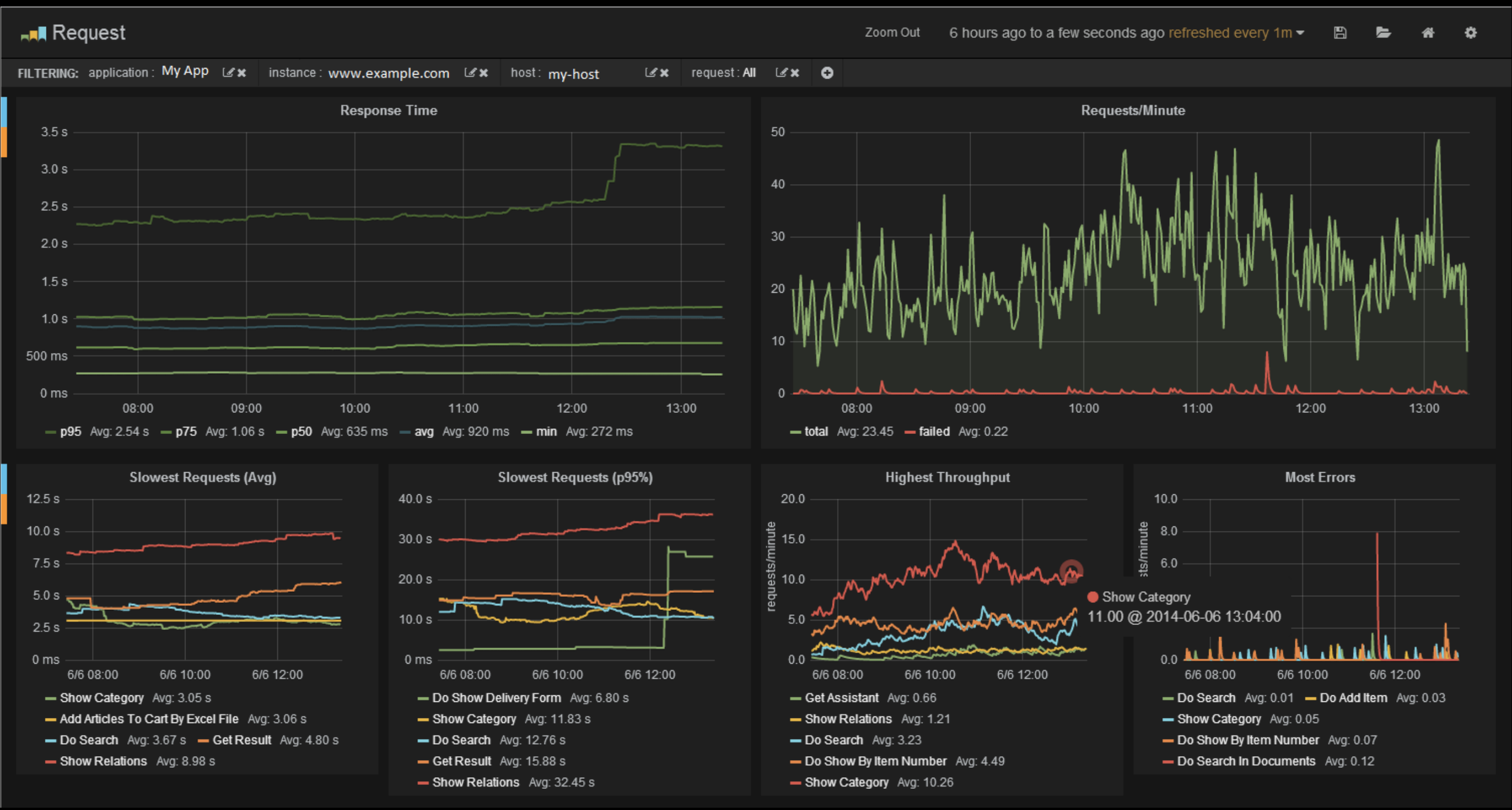
```scala
class Actor extends Actor with Instrumented {
  val meter = metrics.meter("requests")

  def receive: Receive = {

    case Request =>
      meter.mark()

  }
}
```

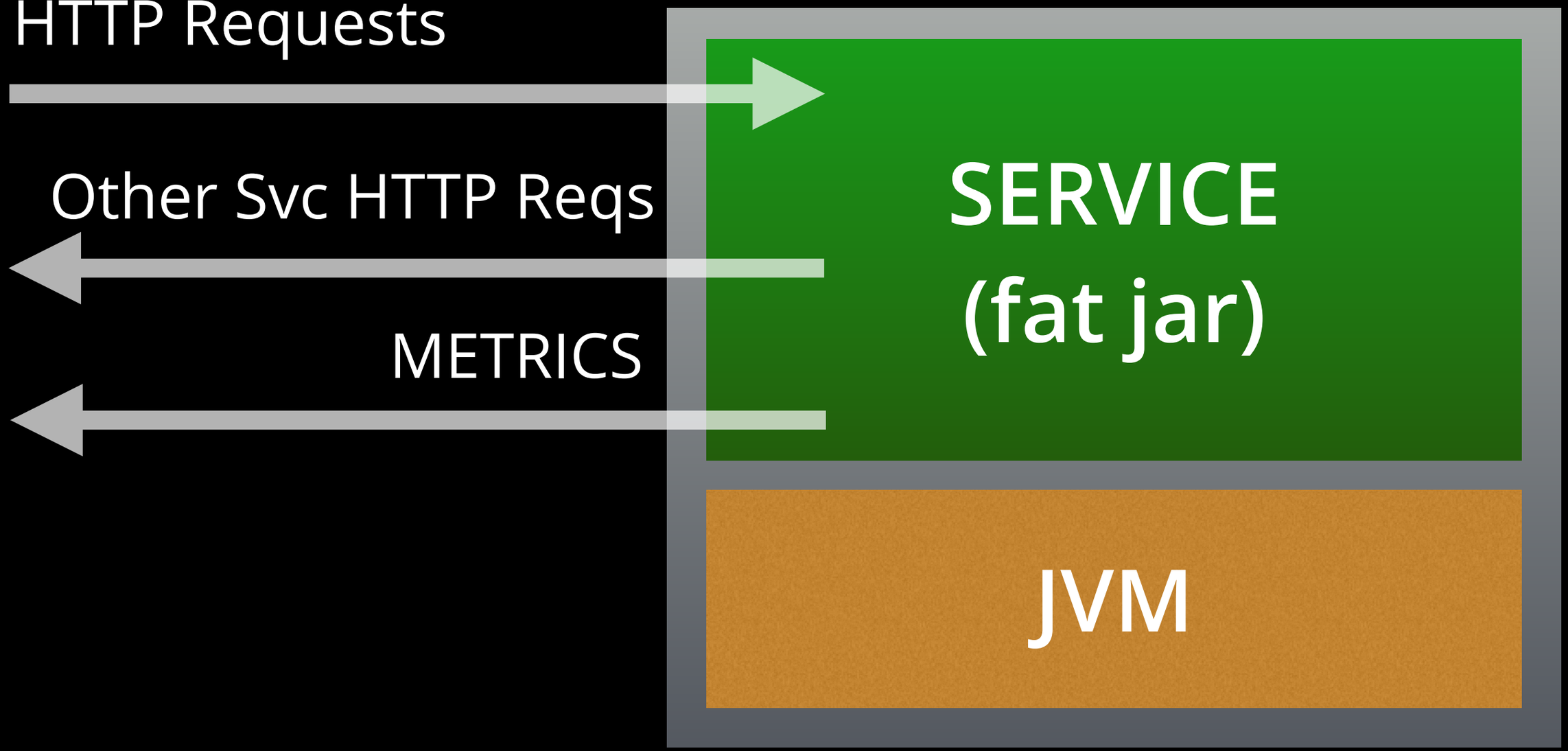
PERFMON = METRICS + GRAPHITE



RELEASING = SBT-RELEASE

- ✓ SNAPSHOT/RELEASE artifacts
- ✓ Semantic versioning
- ✓ version.sbt + Git tags
- ✓ publishing of artifacts
- ✓ 100% customizable process
- ✓ interactive OR automated

PACKAGING = ASSEMBLY + DOCKER



PACKAGING = SBT-ASSEMBLY + SBT-DOCKER

```
docker <<= (docker dependsOn assembly)

dockerfile in docker := {
  val artifact = (outputPath in assembly).value
  val artifactTargetPath = s"/app/${artifact.name}"
  new Dockerfile {
    from(“acme/javabase")
    expose(9999)
    add(artifact, artifactTargetPath)
    entryPoint("java", "-jar", artifactTargetPath)
  }
}

imageNames in docker := Seq(
  ImageName(
    namespace = Some(organization.value),
    repository = name.value,
    tag = Some("v" + version.value)
  )
)
```

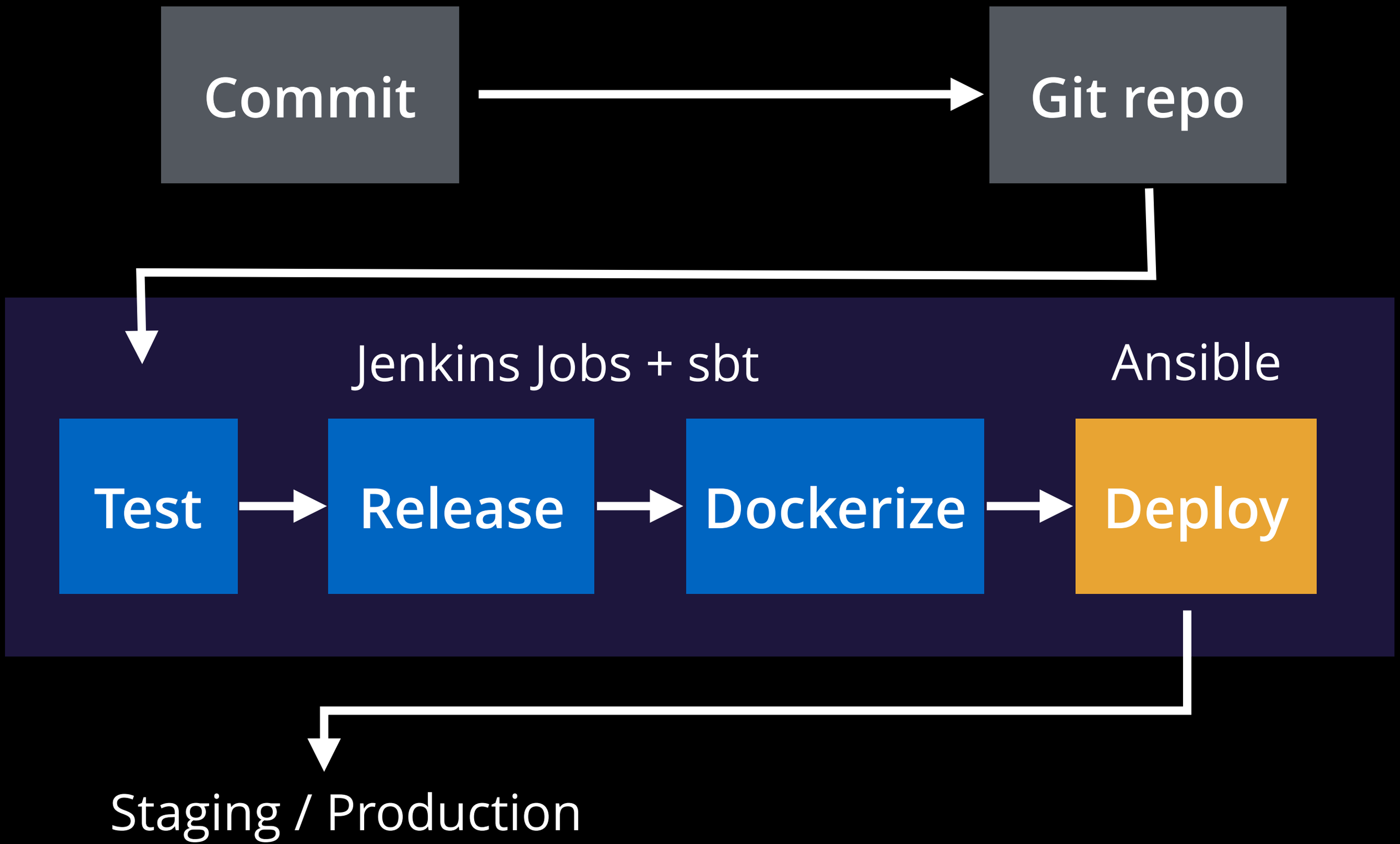
MINIMUM VIABLE SERVICE

- ✓ Business Logic = **Akka**
- ✓ REST API = **Spray + Swagger**
- ✓ Configuration = **Typesafe Config**
- ✓ Health / Monitoring = **Metrics**
- ✓ Packaging = **Assembly & Docker**

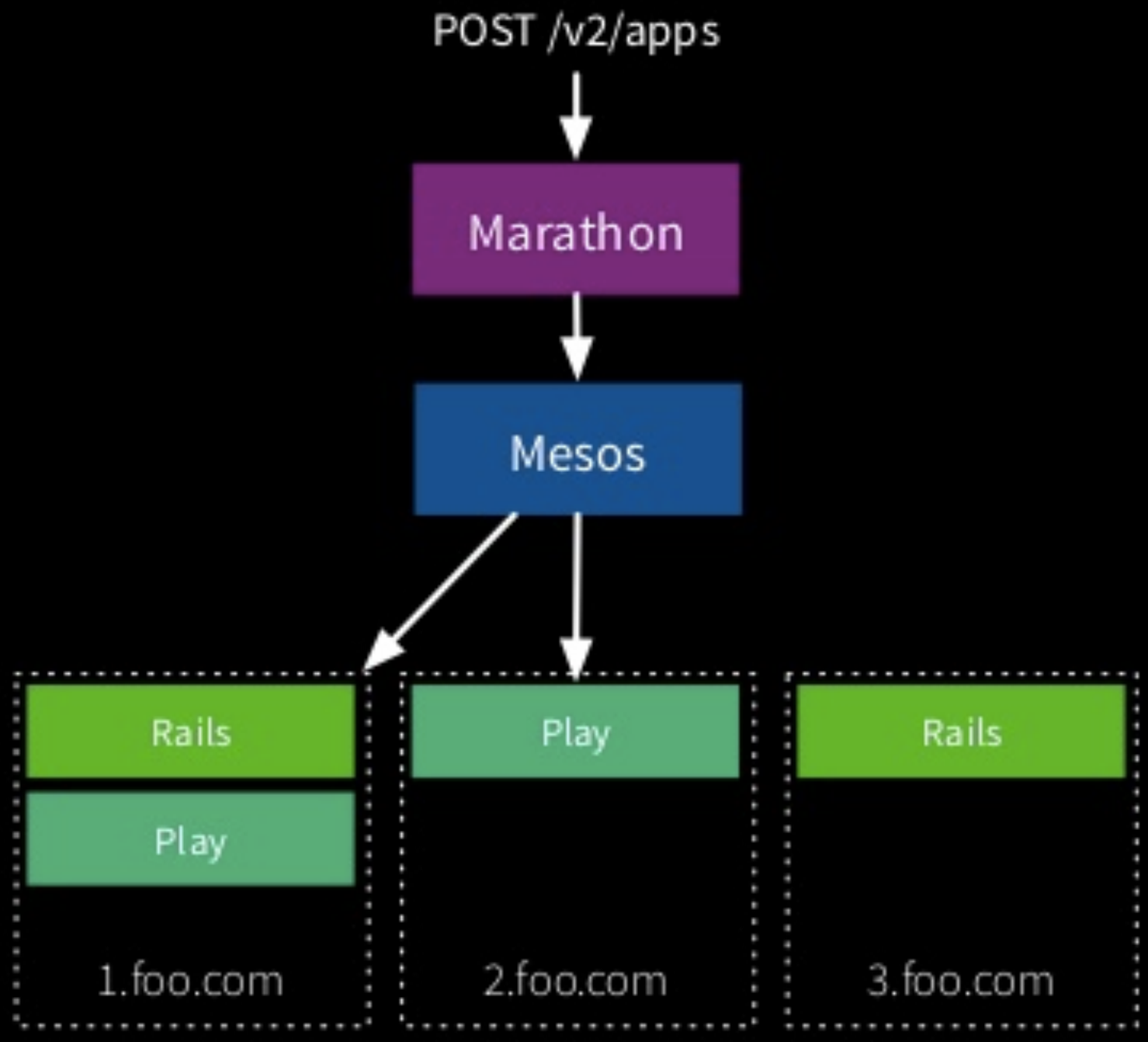
DEPLOYMENT & DISCOVERY

- Continuous Integration
- Deployment
- Service Discovery

CI/DEPLOY = JENKINS + SBT + ANSIBLE



HOSTING = MESOS/MARATHON



mesosphere

HOSTING = MESOS/MARATHON

```
{
    "id": "/svc-calculator",
    "container": {
        "type": "DOCKER",
        "docker": {
            "image": "docker.com/svc-calculator:1.0.0",
            "network": "BRIDGE",
            "portMappings": [{
                "containerPort": 9999, "protocol": "tcp"
            }]
        }
    },
    "env": {"JAVA_ARGS": “-Dconfig.resource=production"},
    "cpus": 1,
    "mem": 1024,
    "instances": 2
}
```

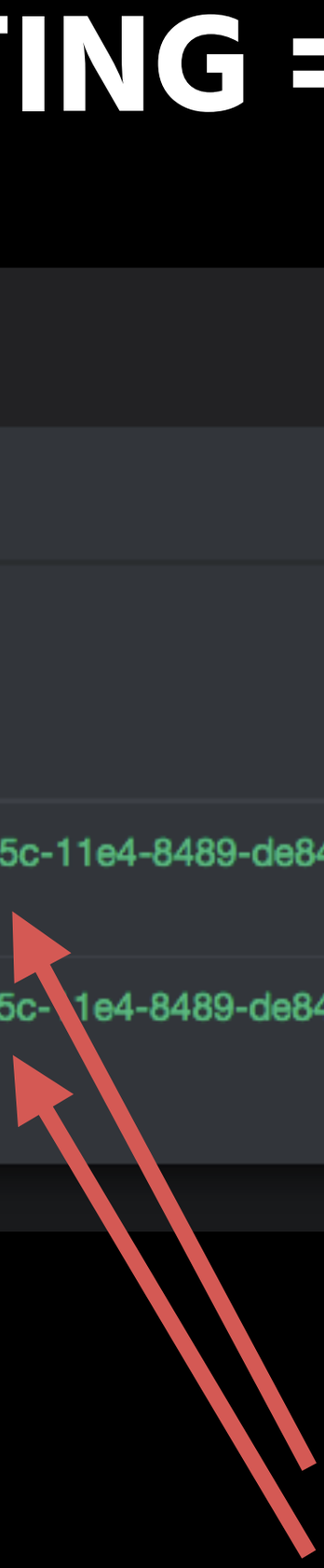
HOSTING = MESOS/MARATHON

Suspend | Scale | Destroy App

Tasks | Configuration

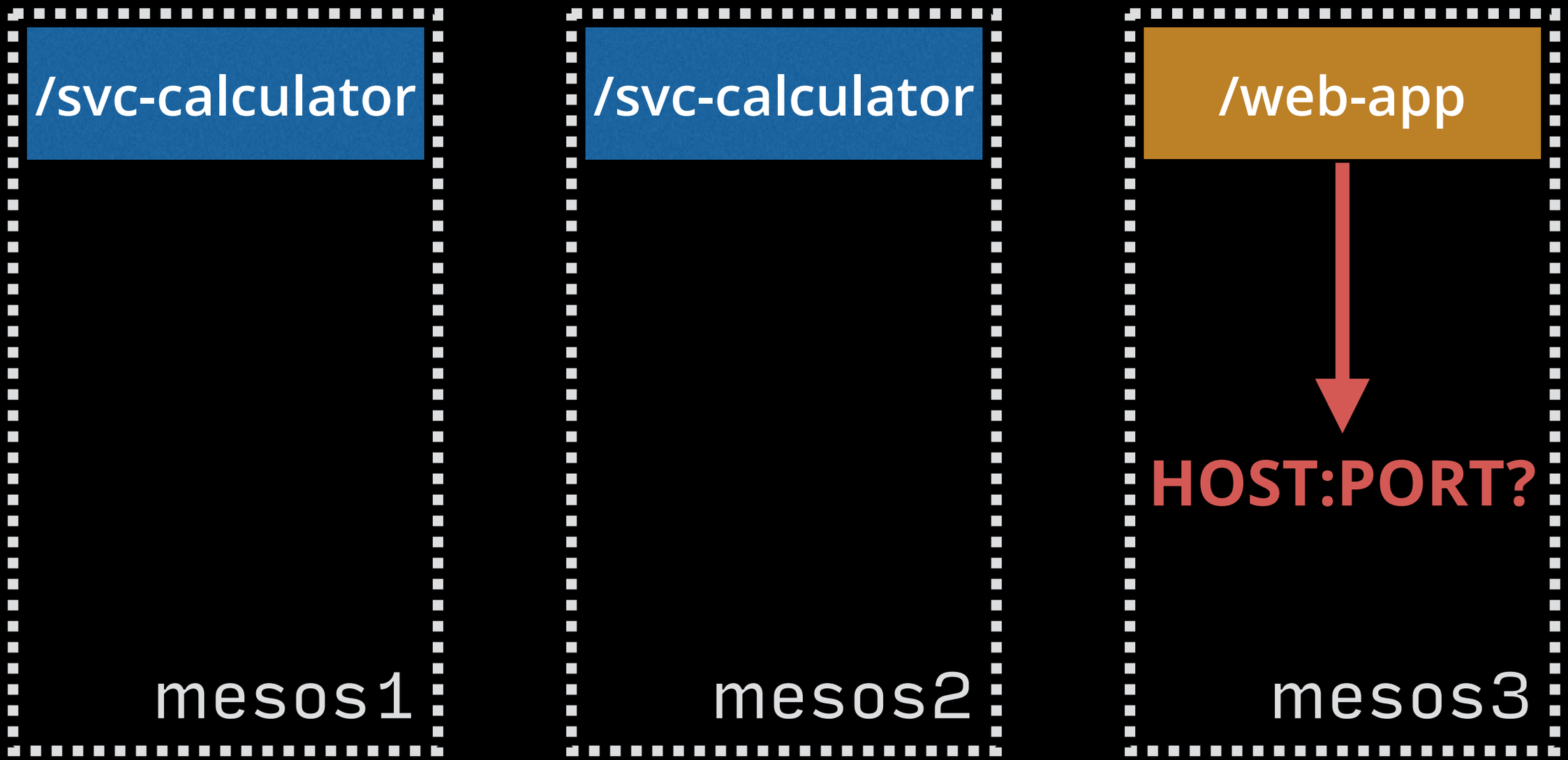
Refresh

ID	Status	Version	Updated
webapp.762bd8dc-7b5c-11e4-8489-de84d2d6cff1 104.131.76.60:31761	Started	12/3/2014, 6:21:30 PM	12/3/2014, 6:28:53 PM
webapp.6d38e478-7b5c-11e4-8489-de84d2d6cff1 104.131.76.60:31705	Started	12/3/2014, 6:21:30 PM	12/3/2014, 6:28:53 PM

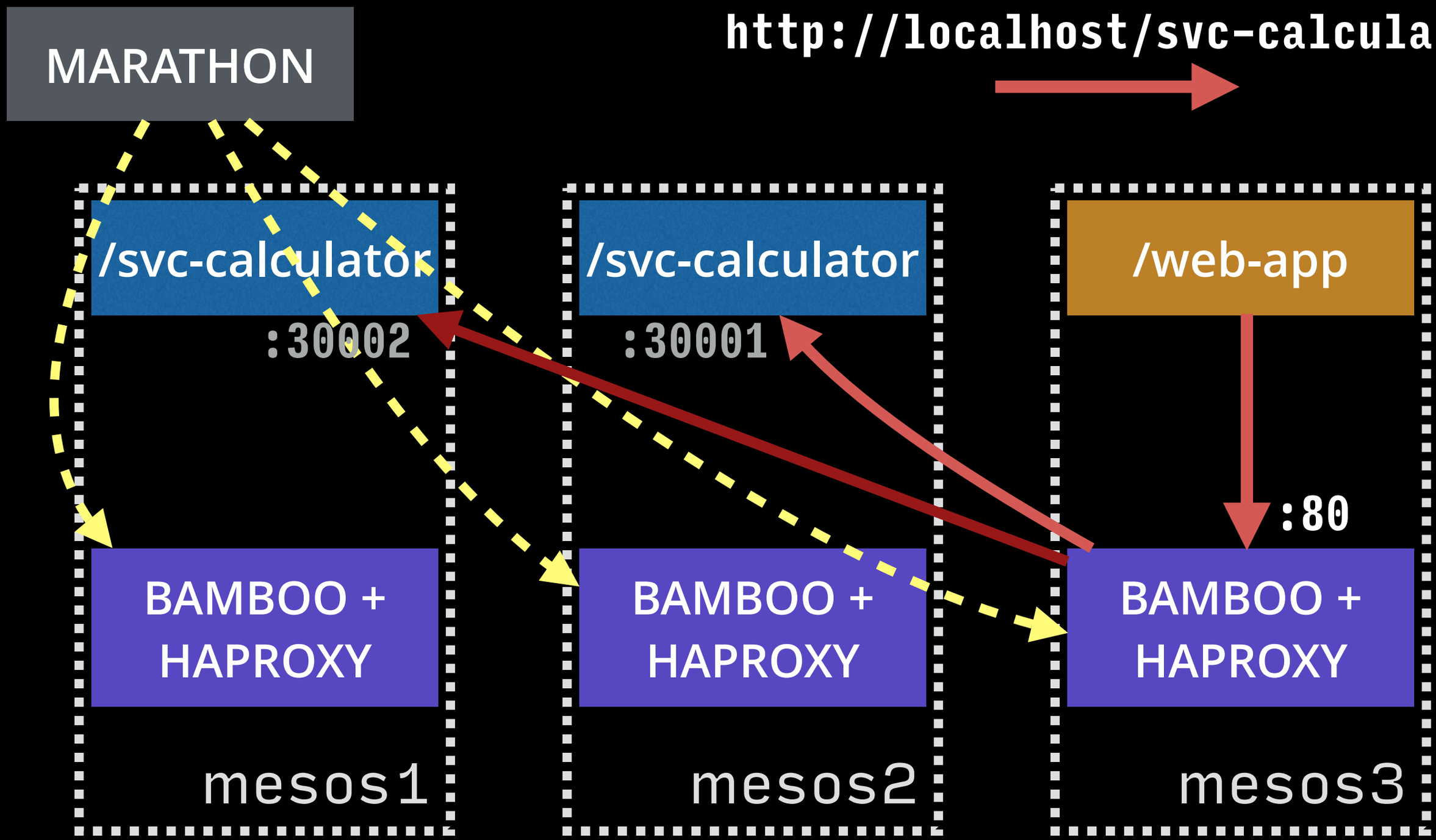


Dynamic port mapping

DISCOVERY = ?


/svc-calculator
/svc-calculator
/web-app
HOST:PORT?
mesos1
mesos2
mesos3

DISCOVERY = BAMBOO + HAPROXY



DEPLOYMENT & DISCOVERY

- ✓ CI = **Jenkins + sbt**
- ✓ Deploy = **Ansible + Marathon**
- ✓ Discovery = **Bamboo + HAProxy**

FINAL TIPS

- ✓ DRY + keep team aligned
- ✓ Zero friction on new services

DRY: CUSTOM SBT PLUGIN

```
package com.acme.sbt.plugin
import sbt._
import Keys._
object AcmePlugin extends AutoPlugin {
  lazy val acmeBuildSettings = Seq(
    scalacOptions ++= Seq(
      "-unchecked",
      "-deprecation",
      "-feature",
      "-Xlint",
      "-Ywarn-dead-code",
      "-target:jvm-1.7"
    )
  )
}
```

AcmePlugin.scala

FINAL TIP: CUSTOM SBT PLUGIN

```
import com.acme.sbt.plugin.AcmePlugin._

Seq(acmeBuildSettings:_*)

// custom project settings
```

build.sbt

ZERO FRICTION: GITER8 TEMPLATES

```
% g8 file://g8-svc-spray/ --name=svc-calculator \
    --servicePort=9999
Template applied in ./svc-calculator

% ls svc-calculator
build.sbt   project     src         version.sbt
```

Recap


ONE DOES NOT
SIMPLY
DEPLOY TO PRODUCTION
memegenerator.net


DEPLOY
SUCCESSFULLY DONE!

Thank you!

WE’RE HIRING! **JOBS@MEASURENCE.COM**