ThoughtWorks®

12 Factors App with Docker On AWS

ThoughtWorks®

About Me : Ma Bowen

- ThoughtWorks Senior Consultant
- Web/RoR/Java/Scala Developer, 3 years DevOps
- Book Translation <Scala Cookbook>
- AWS Certified Associate Solution Architect

ThoughtWorks®

12facters.net

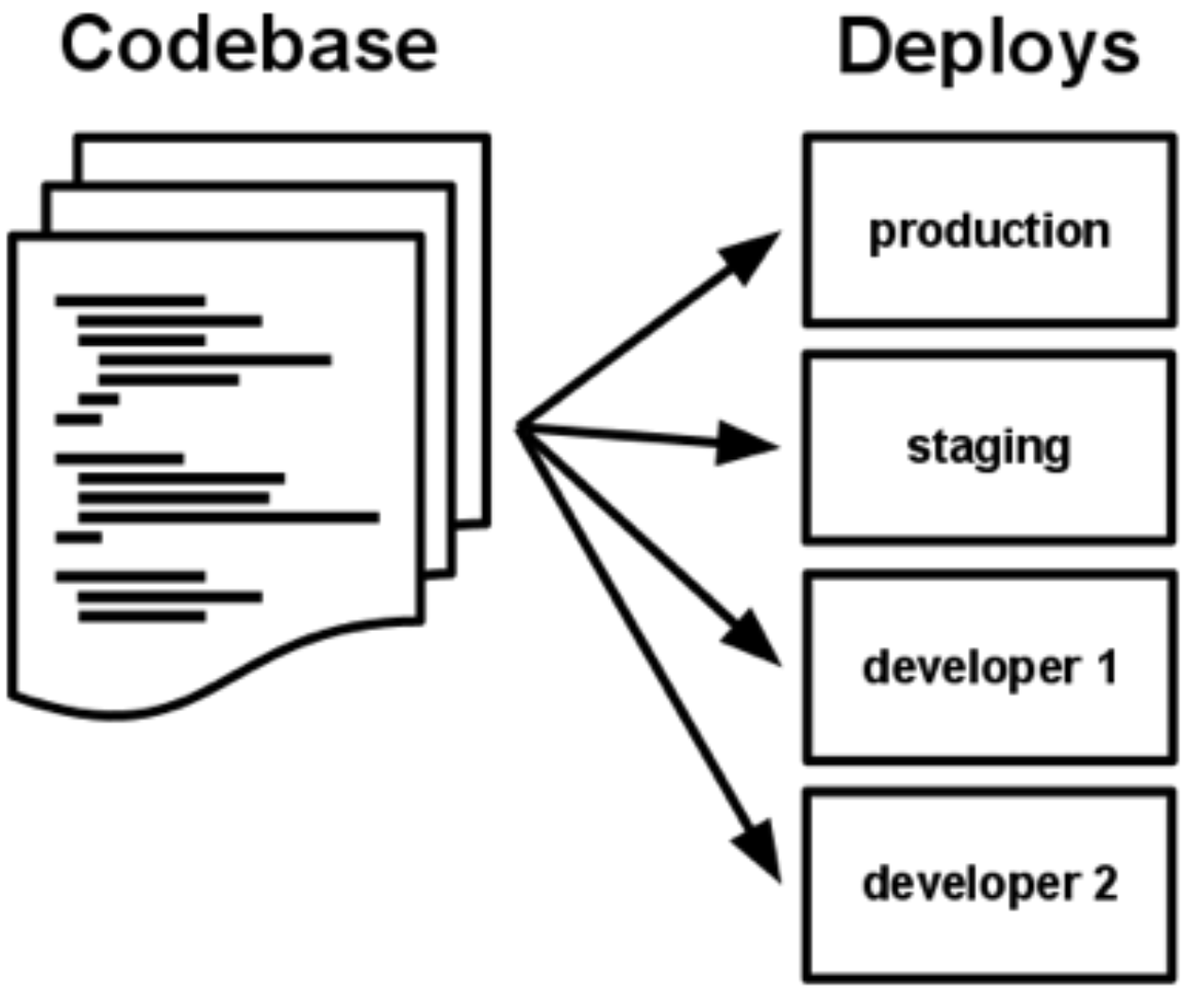
ThoughtWorks®

Methodology for building Web Apps

- Use declarative formats for setup automation
- Maximum portability between execution environments
- Suitable for deployment on modern cloud platforms
- keep environment consistence, continuous deployment
- Scale with few changes to tooling/architecture etc

ThoughtWorks®

One Codebase, Multiple Deploys



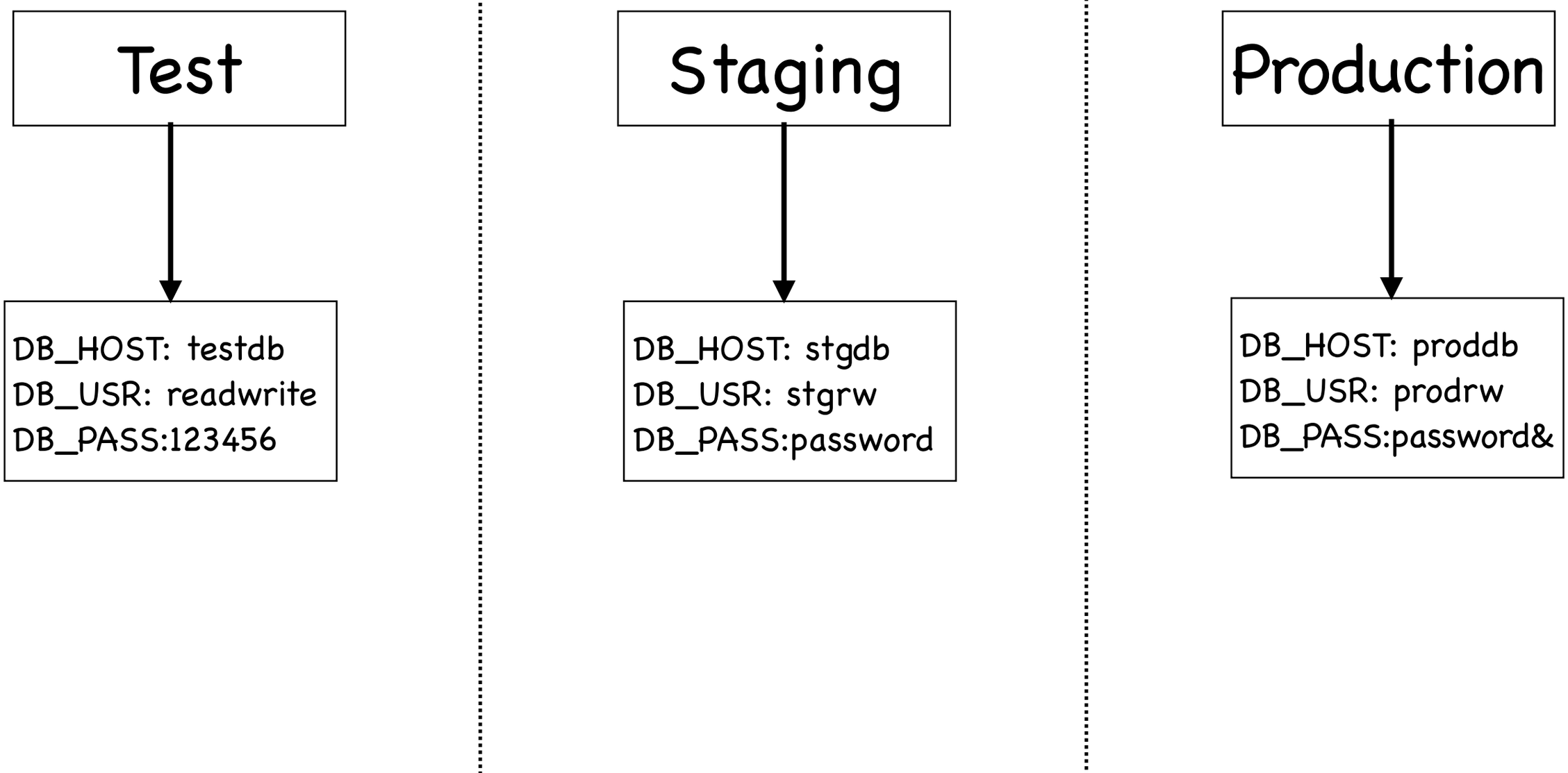
ThoughtWorks®

Explicitly declare & isolate dependencies

- Ruby Gemfile, e.g `bundle install —path=vendor/bundle`
- Debian/RPM

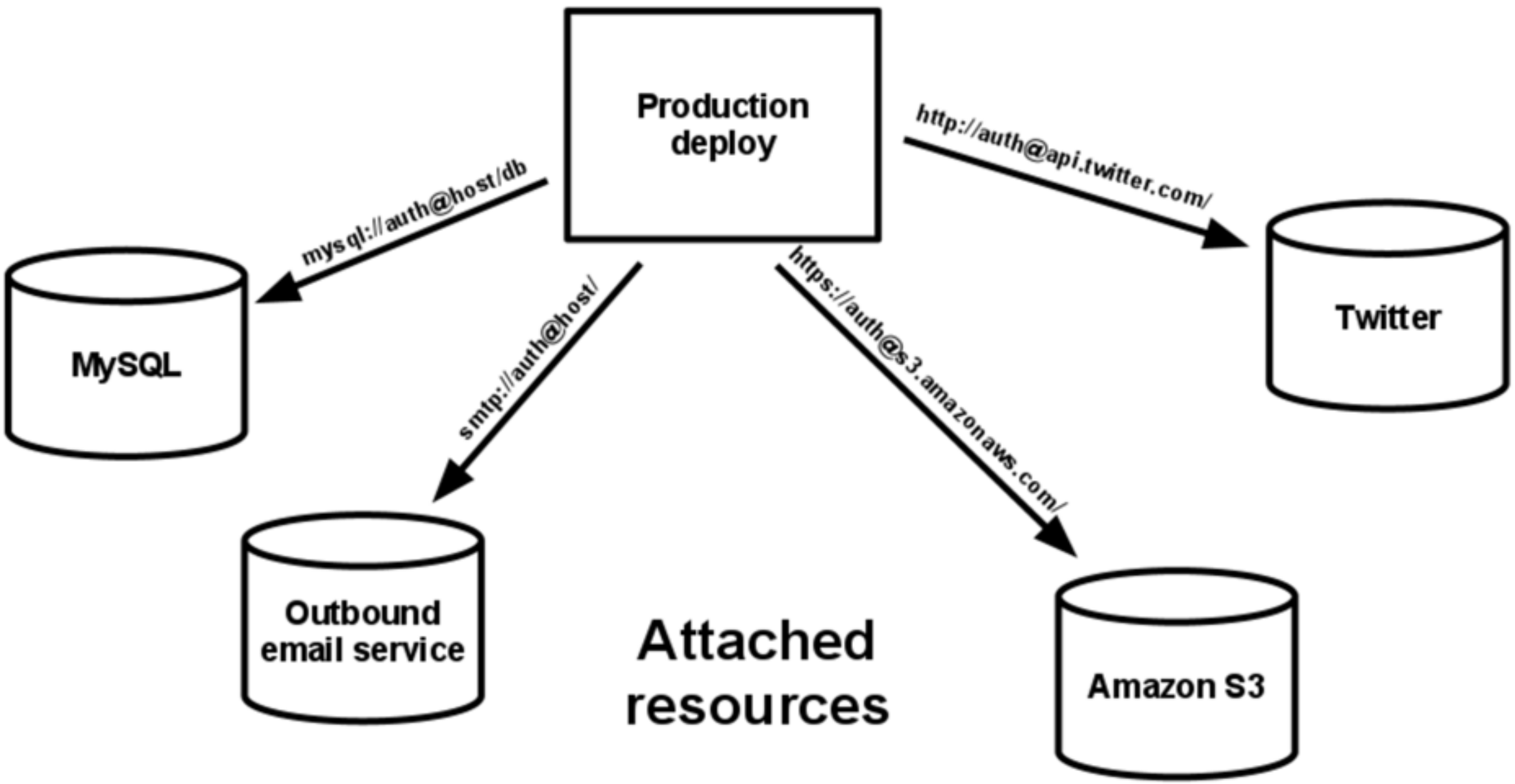
ThoughtWorks®

Store Config in Environment



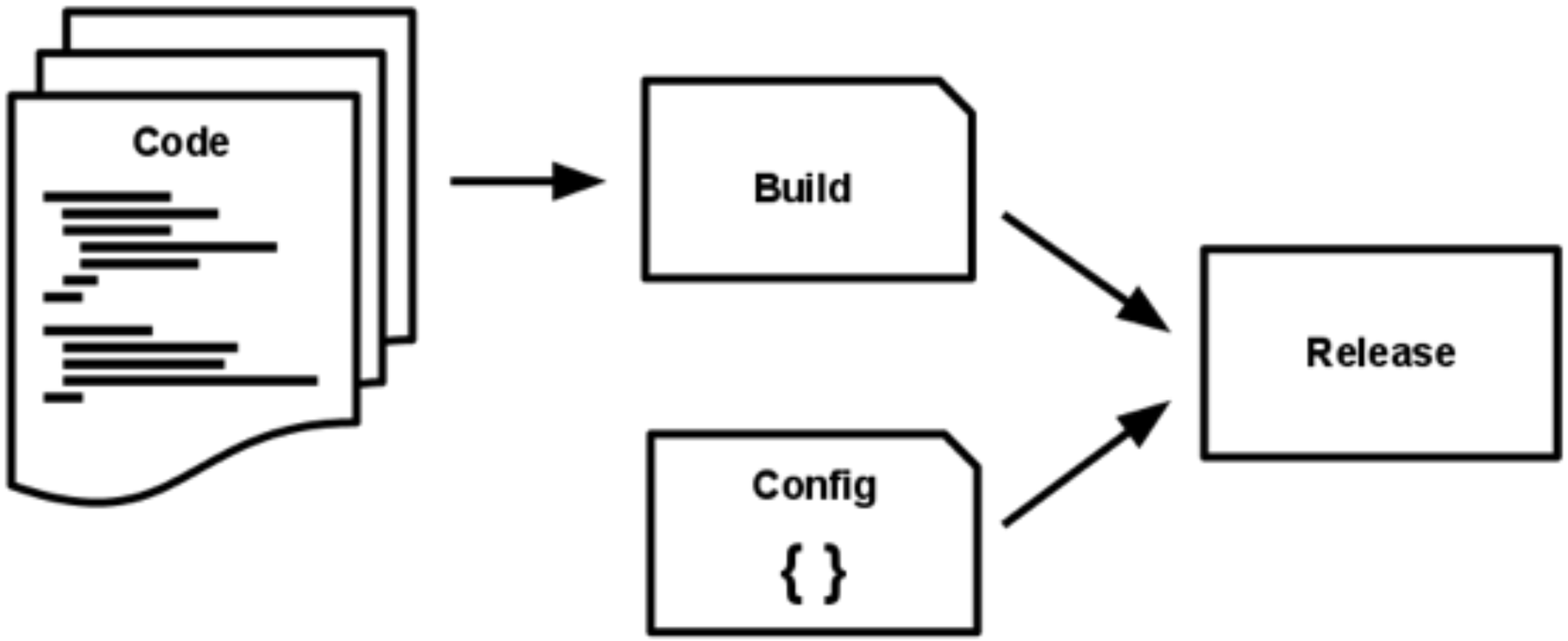
ThoughtWorks®

Backing services as attached resources



ThoughtWorks®

Build release run



ThoughtWorks®

Processes

The app is executed in the execution environment as one or more processes.

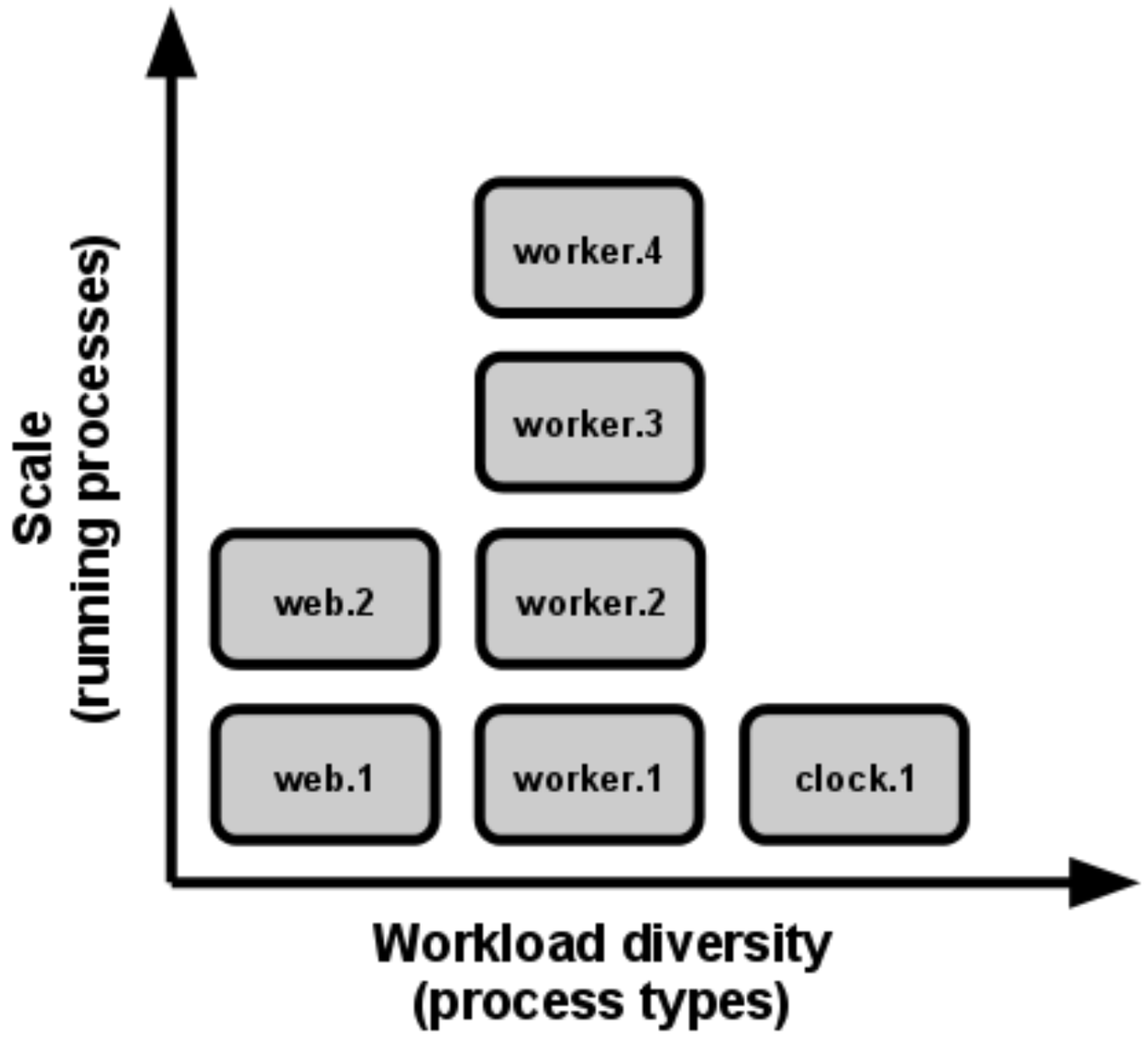
Twelve-factor processes are stateless and share-nothing

Port Binding

The twelve-factor app is completely self-contained and does not rely on runtime injection of a webserver into the execution environment to create a web-facing service.

ThoughtWorks®

Concurrency



Twelve-factor app processes should never daemonize or write PID files. Instead, rely on the operating system's process manager (such as Upstart, a distributed process manager on a cloud

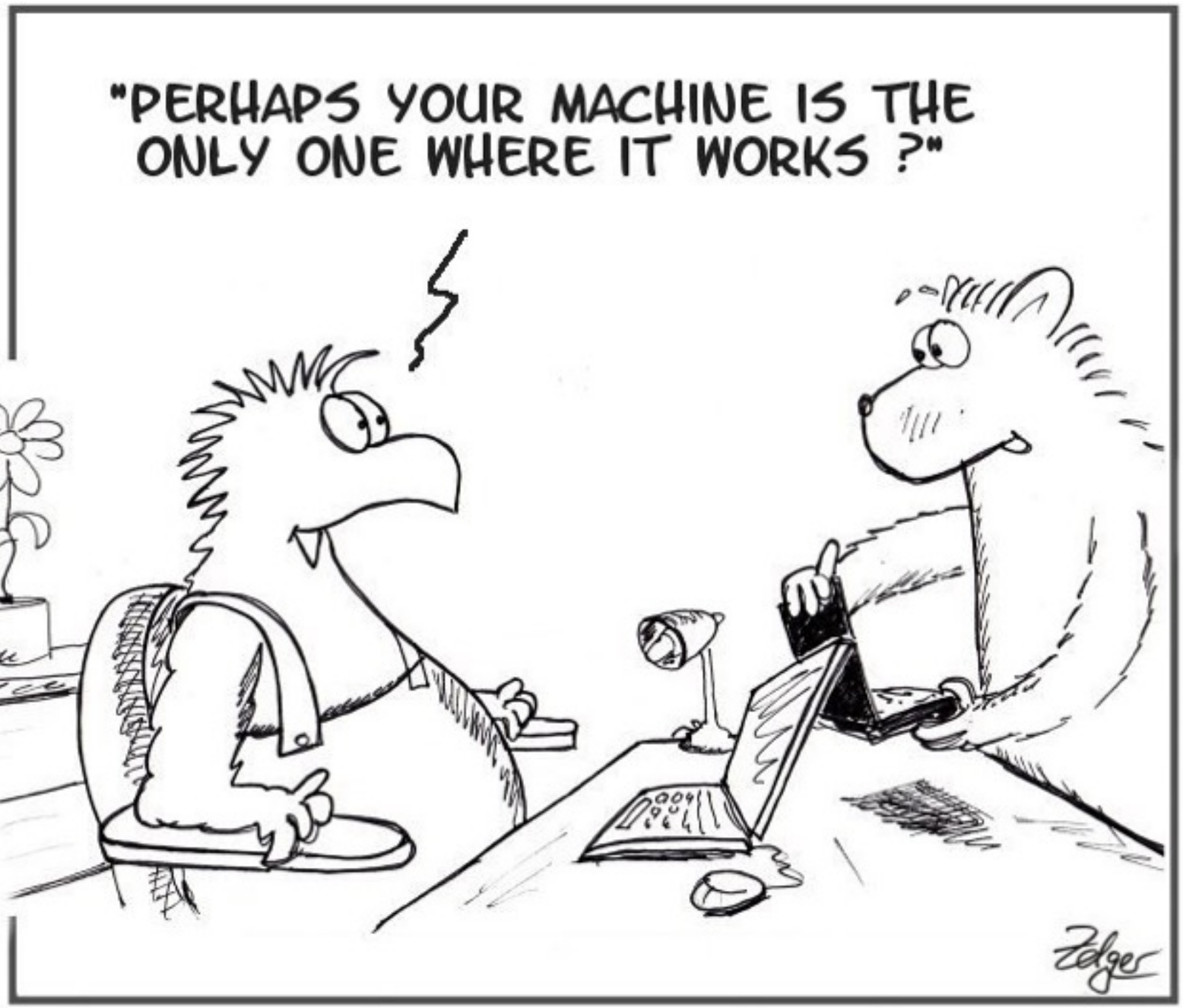
ThoughtWorks®

Disposability

Maximize robustness with fast startup and graceful shutdown

ThoughtWorks®

Dev/prod parity



It works on my machine

ThoughtWorks®

Logs

Treat logs as event streams,A twelve-factor app never concerns itself with routing or storage of its output stream.

ThoughtWorks®

Admin processes

ASG

Scheduled action

Instance

One Off task

Run admin/management tasks as one-off processes

ThoughtWorks®

How do we apply this on AWS with Docker

ThoughtWorks®

Some Context

Years Ago	Now
Dev | Ops	DevOps,Cross Functional team
8 teams	40+ teams
monoliths	micro services(decomissioning)
2 Data Centers	2 DC + 100 AWS accounts
Ops Deploying	TMI && Continuous Delivery

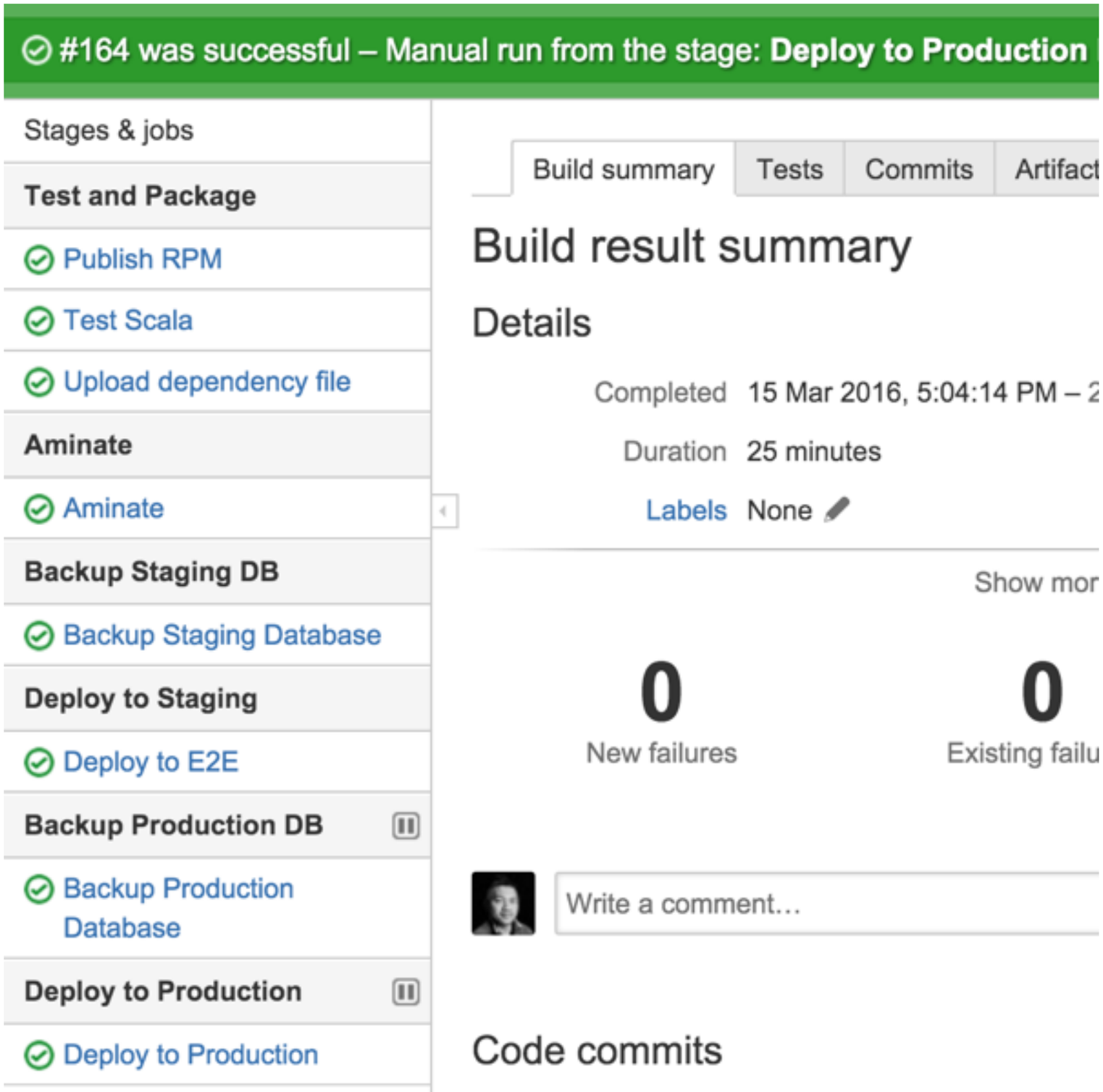
ThoughtWorks®

Some Glossaries

- AMI: Amazon Machine Image
- ELB: Elastic Load Balancer
- ASG: Auto Scaling Group
- Cloudwatch: AWS Monitoring Service
- CloudFormation: Manage AWS resources with JSON template
- Newrelic: Application Monitoring
- Splunk: Enterprise Log Aggregator

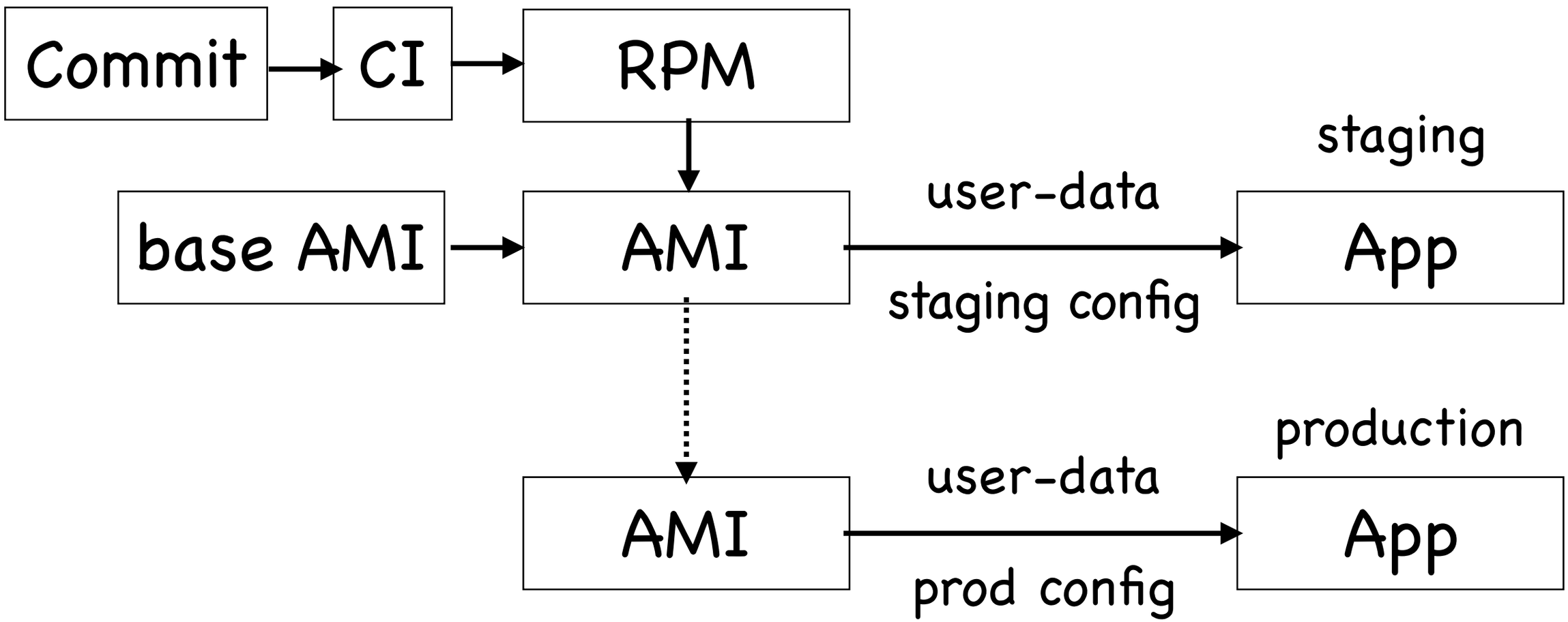
ThoughtWorks®

Continuous Delivery Before



ThoughtWorks®

Processes



ThoughtWorks®

Cons

- Packaging twice, RPM/AMI
- Duplicated effort for automate deployment
- Not good for succession plan
- Dev/Test/Staging/Prod different

ThoughtWorks®

Docker - FTW

ThoughtWorks®

Expected

ThoughtWorks®

AIM of shipper

- Standardising and simplify the way we deploy
- Portable between teams and account

ThoughtWorks®

shipper.yml

```yaml
app:
  name: app
  image: app
  environment:
    SOME_ENV: "some_env"
  health_check:
    path: /diagnostic/status/heartbeat
  port: 9090

aws:
  instances:
    type: t2.micro
  load_balancer:
    scheme: internet-facing
    listeners:
      443:
        to: 80
        protocol: https
splunk:
  host: splunk
  index: app
```

ThoughtWorks®

Any web application
any web-app framework
any programming language
any Linux variant

Auto-scaling
multiple servers
load-balancing
support for scaling schedules

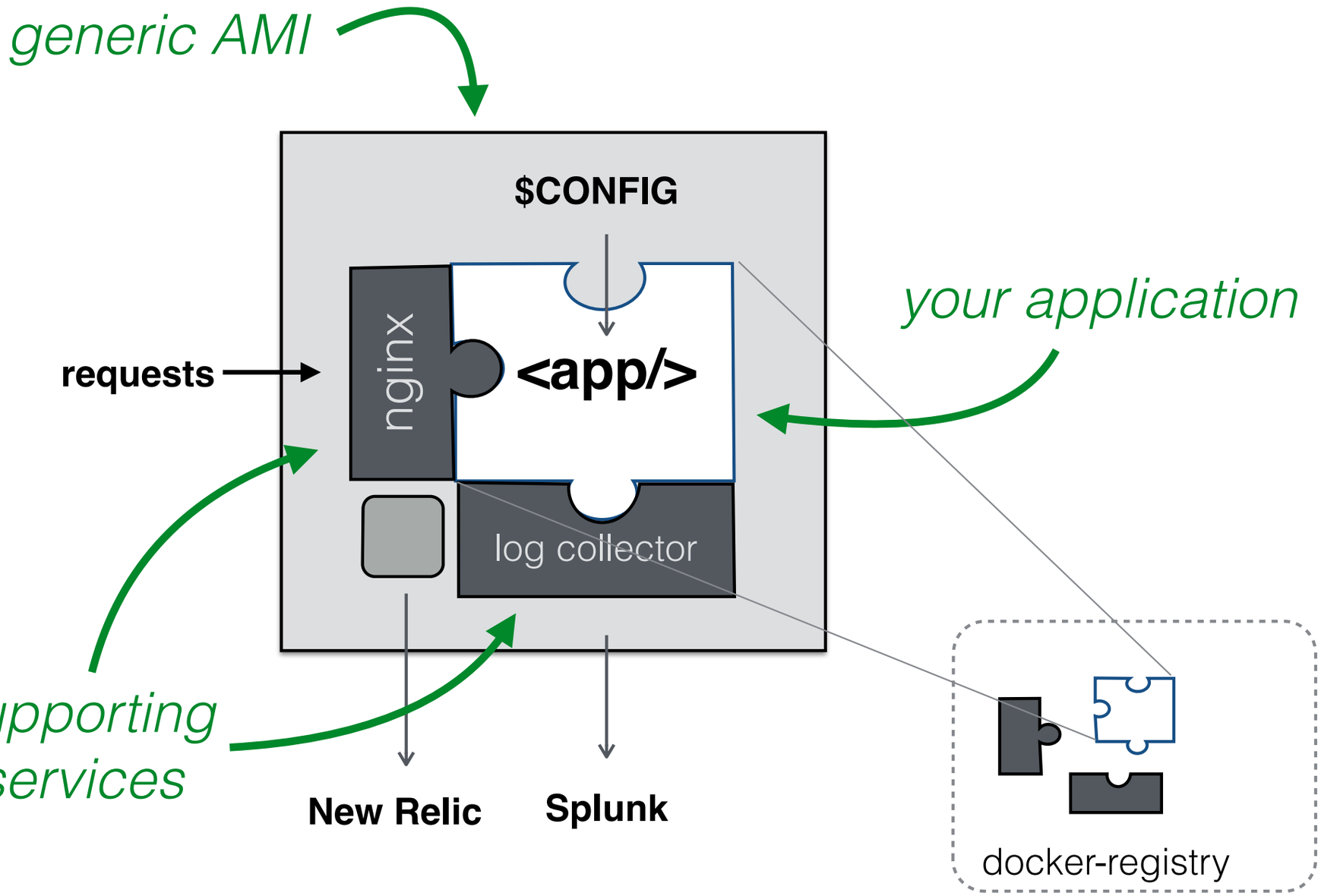
Splunk support
captures stdout/stderr
no app support required

Zero-downtime deployments
safe upgrades
safe config changes

CloudWatch support
alerts you when service is down
via email or web-hook

New Relic support
application monitoring
system monitoring
deployment notification

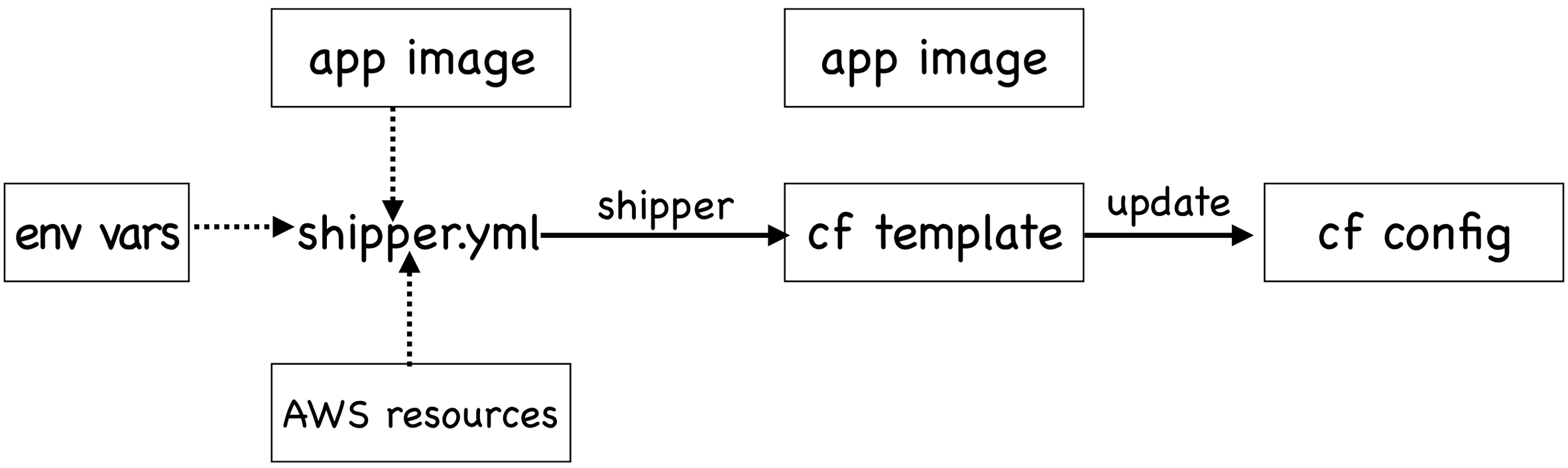
ThoughtWorks®



ThoughtWorks®

Actual Process

ThoughtWorks®



ThoughtWorks®

Immutable Deployment (1/2)

ThoughtWorks®

Immutable Deployment (2/2)

ThoughtWorks®

Docker V2 registry

ThoughtWorks®

Deployed 70+ systems

ThoughtWorks®

Next Step

- Support batched jobs
- ECS/ECR
- Swarm/Kubernates

ThoughtWorks®

Fin

iambowen.github.io

iambowen.m@gmail.com