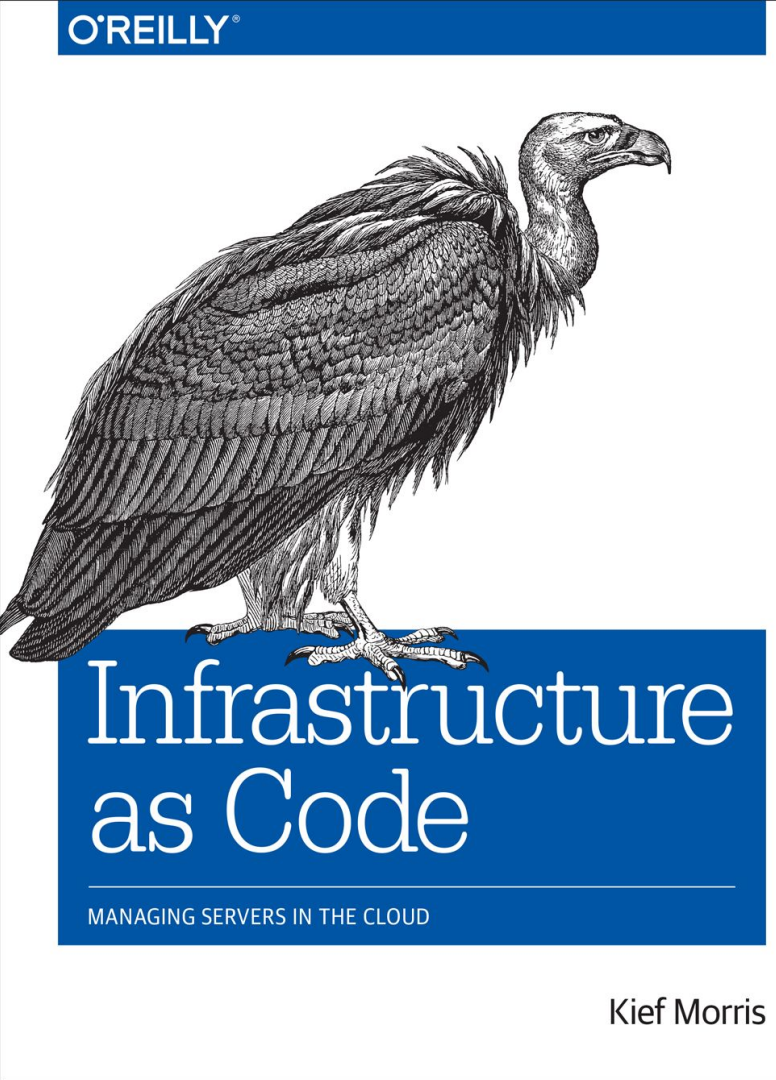

QCon
NEW YORK
INFRASTRUCTURE
AS CODE



kief@thoughtworks.com

Cloud Practice Lead (UK)

DevOps, Continuous Delivery, Agile Ops

ThoughtWorks®

Twitter: @kief
Book: http://oreil.ly/1JKIBVe
Site: http://infrastructure-as-code.com

June 2016

AGENDA

CONTEXT

- Motivations
- Challenges

INFRASTRUCTURE AS CODE

- Key Practices
- Simple Pipeline
- Scaling Pipelines

SPEED

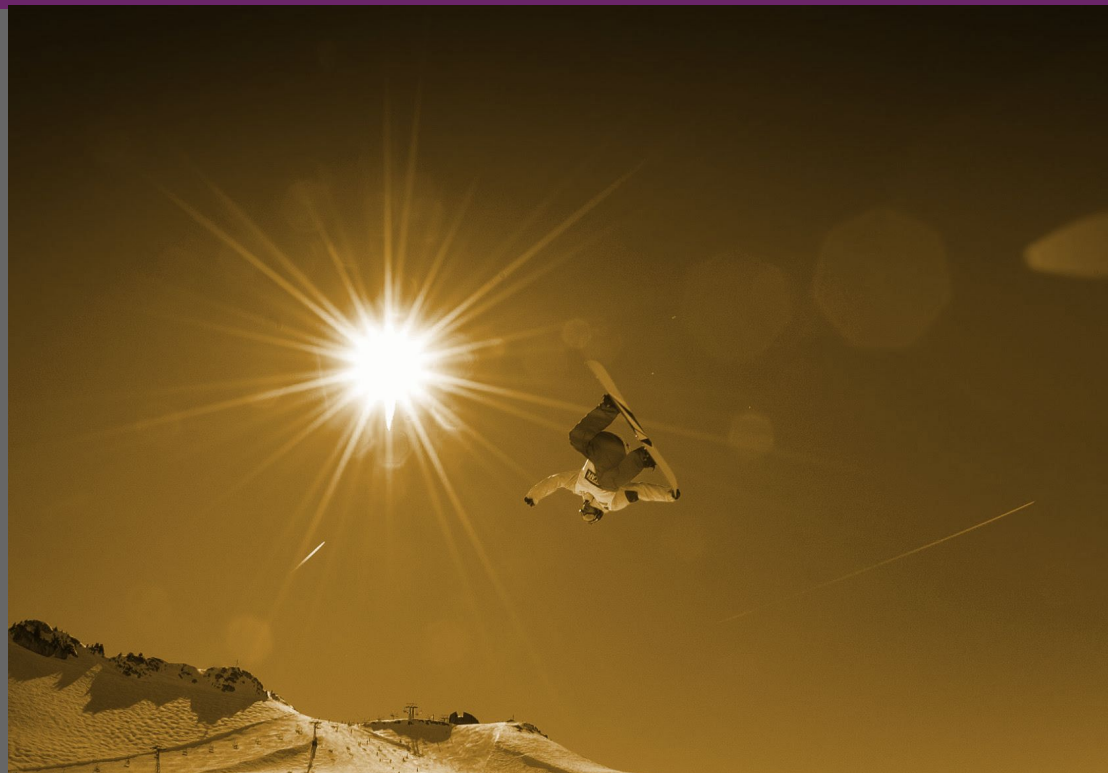
Get something to market **quickly**

Iterate it

Continuously improve it

TECHNOLOGY

Cloud, automation, etc. **lowers barriers** for making changes



DANGER

Security

Performance

Stability

Compliance

Maintainability

SECRET

High quality services rely on the ability to make changes quickly

GOAL

Be able to make changes

Rapidly,

Frequently,

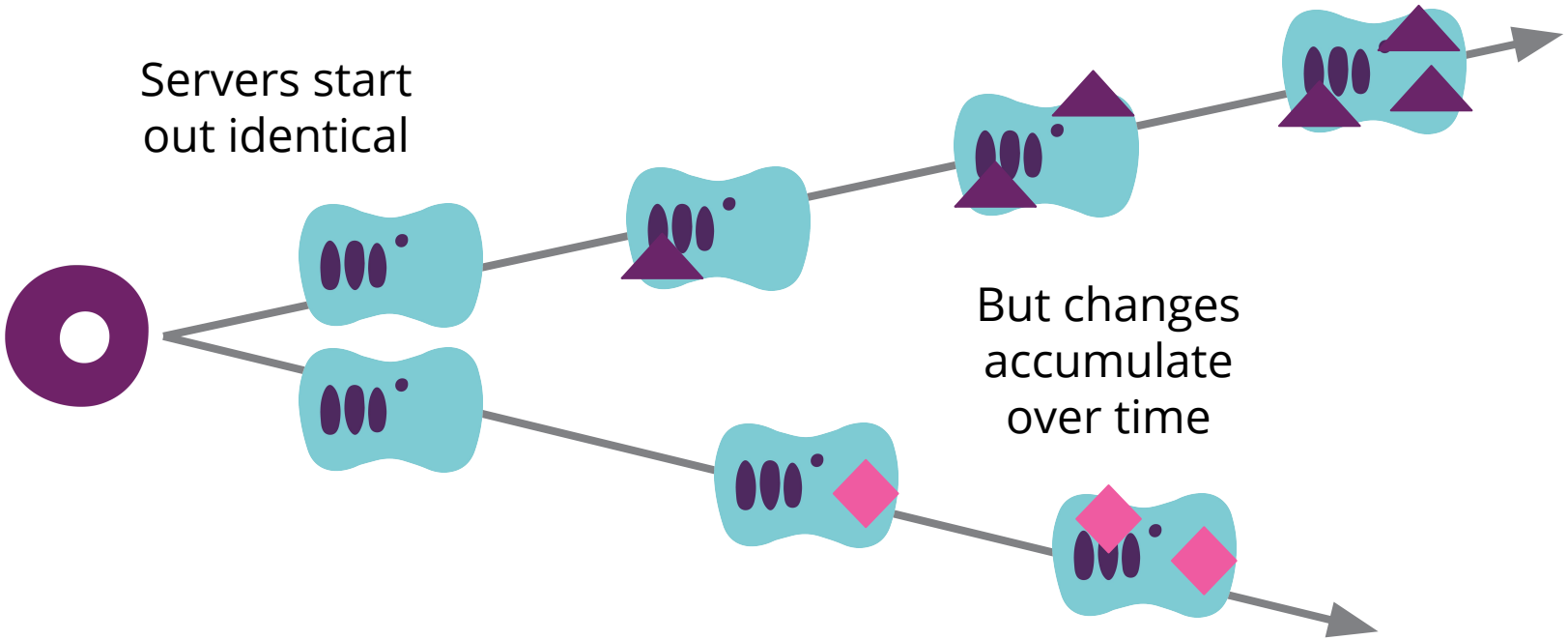
and **Responsibly**

CHALLENGES

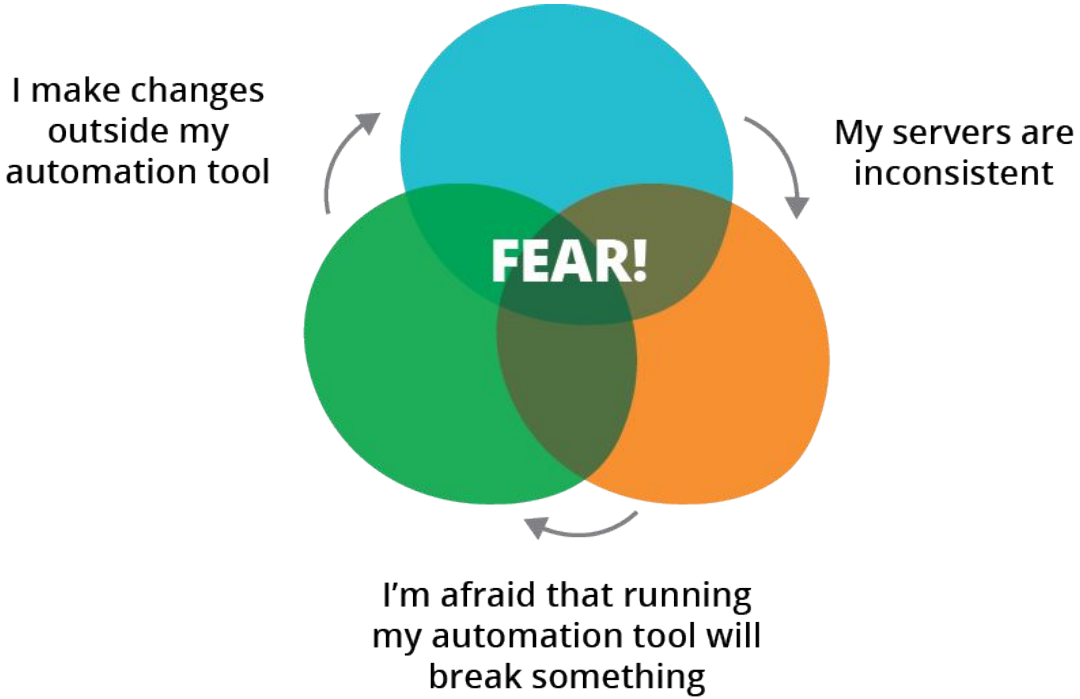
SERVER SPRAWL

Creating new servers is the easy part

CONFIGURATION DRIFT



AUTOMATION FEAR CYCLE



INFRASTRUCTURE AS CODE

“Applying software engineering tools and practices to infrastructure”

UNATTENDED AUTOMATION

Tools run on a schedule to apply, re-apply, and update configuration

BENEFITS OF UNATTENDED:

- **Discover** problems quickly
- Force yourself to **fix** those problems
- Force yourself to **improve** your tools and processes
- Discourages “out of band” changes

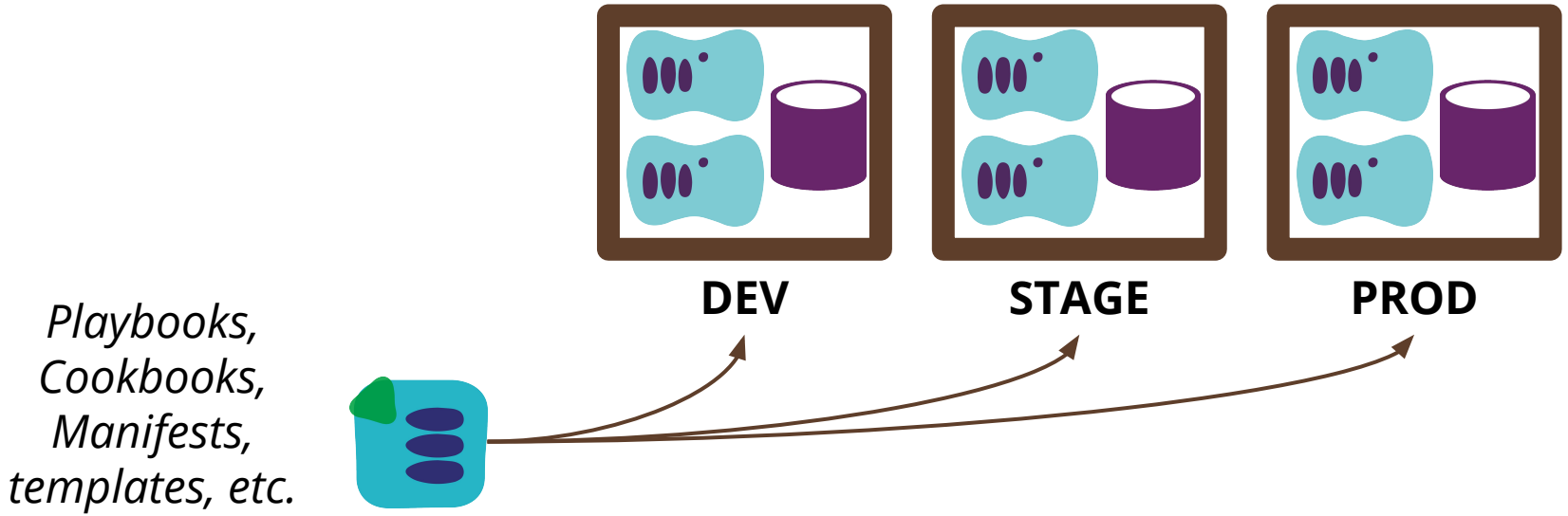
AUTOMATE SERVER UPDATES

Automation isn't just for new servers!

Configuration synchronization	Run Chef, Puppet, Ansible, etc. on a schedule
Immutable servers	Apply changes by rebuilding servers
Containerized servers	Apply changes by deploying new container instances

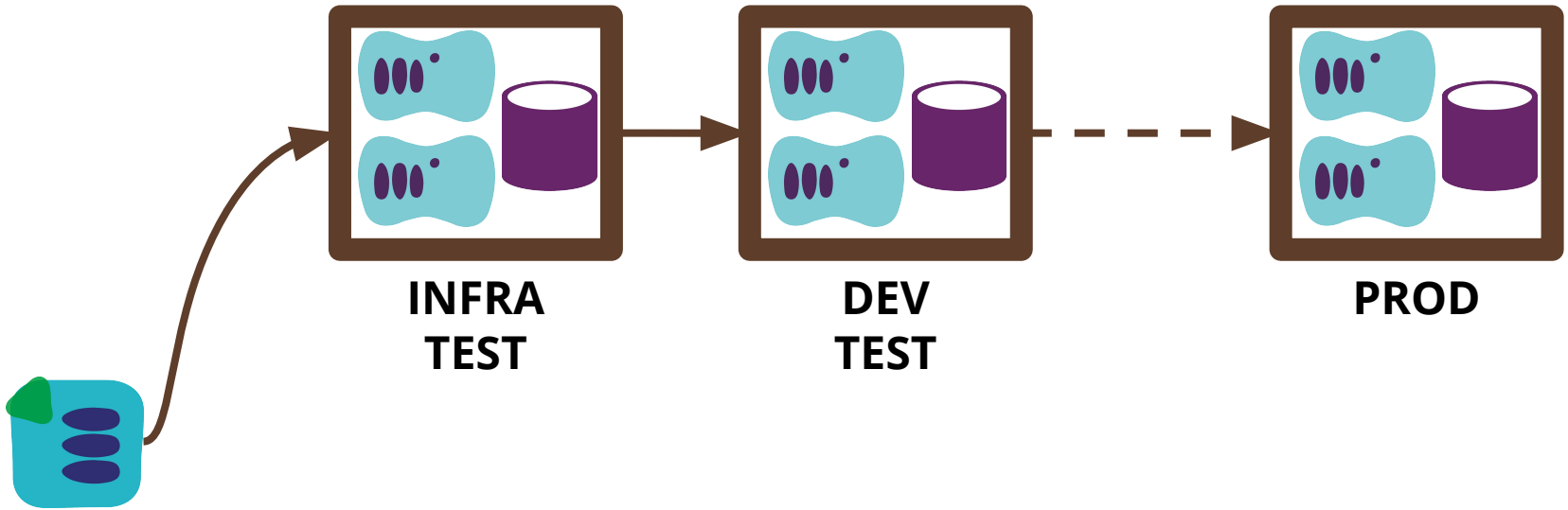
RE-USE & PROMOTE DEFINITIONS

Re-use the same definition files across environments for a given application or service



TEST INFRASTRUCTURE CHANGES

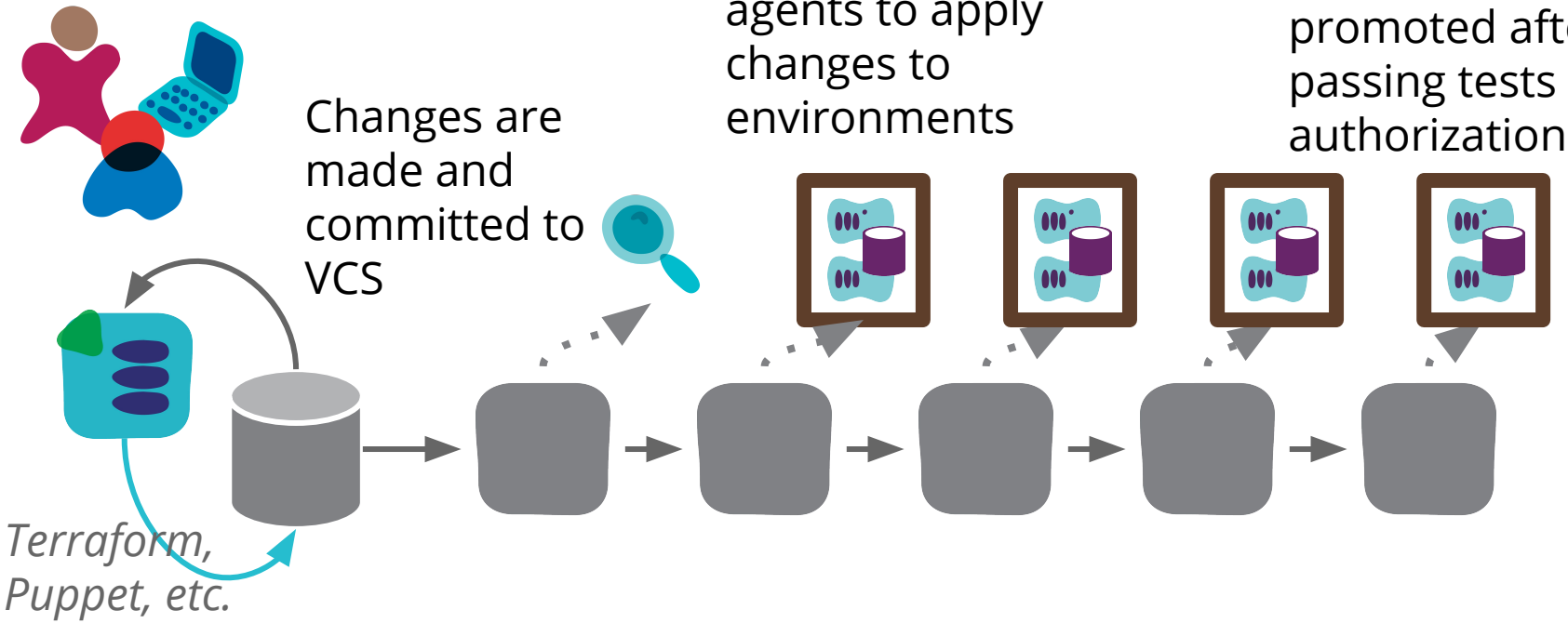
Preventing Dev**Oops**



PIPELINES

Using Continuous Delivery pipelines to manage infrastructure

WHAT?

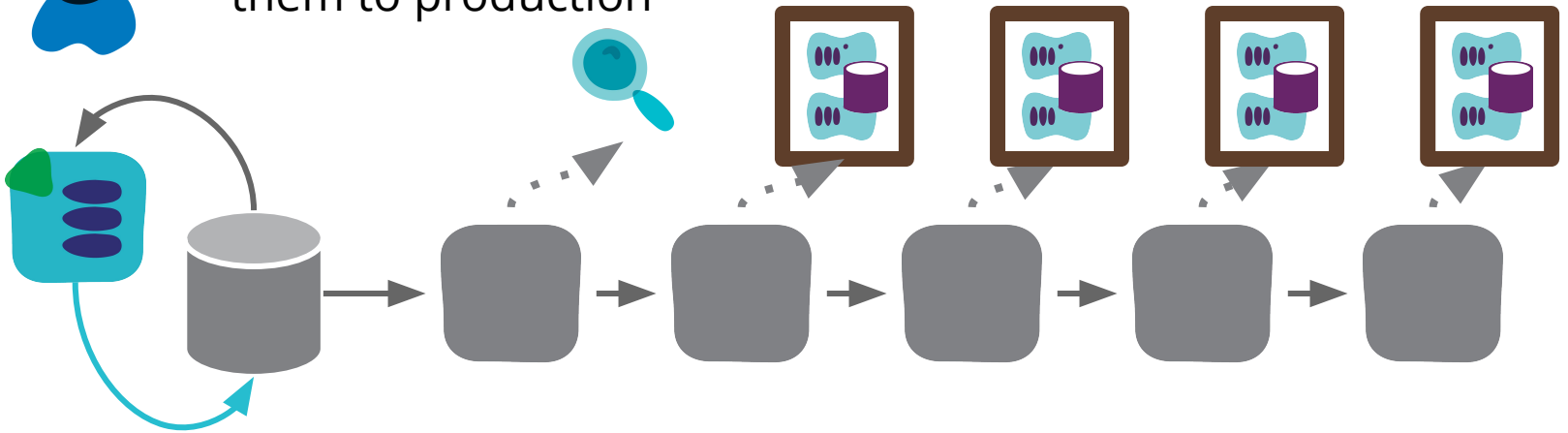

Changes are made and committed to VCS
Tools are run on agents to apply changes to environments
Changes are only promoted after passing tests & authorization
Terraform, Puppet, etc.

WHY?

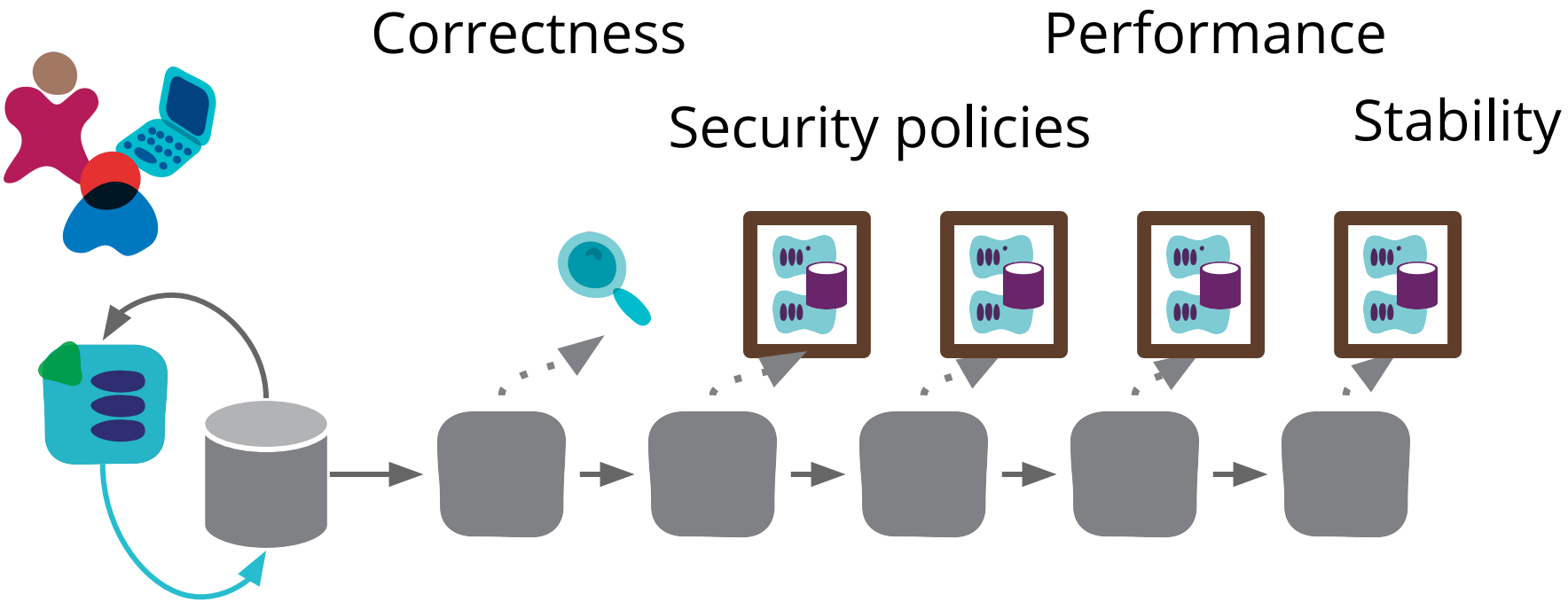
Validates changes to infrastructure before applying them to production

Confidence for frequent, small improvements to infrastructure

Limit direct changes to infrastructure

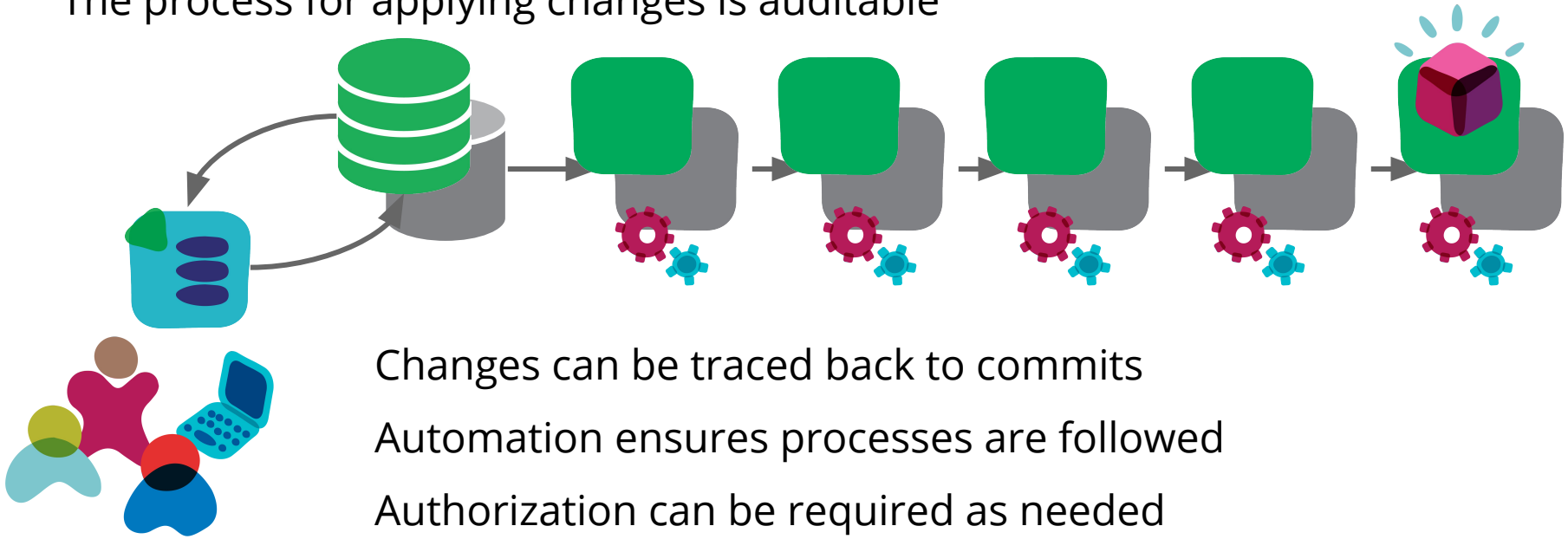


TESTING



GOVERNANCE

The process for applying changes is auditable



Changes can be traced back to commits

Automation ensures processes are followed

Authorization can be required as needed

SIMPLE

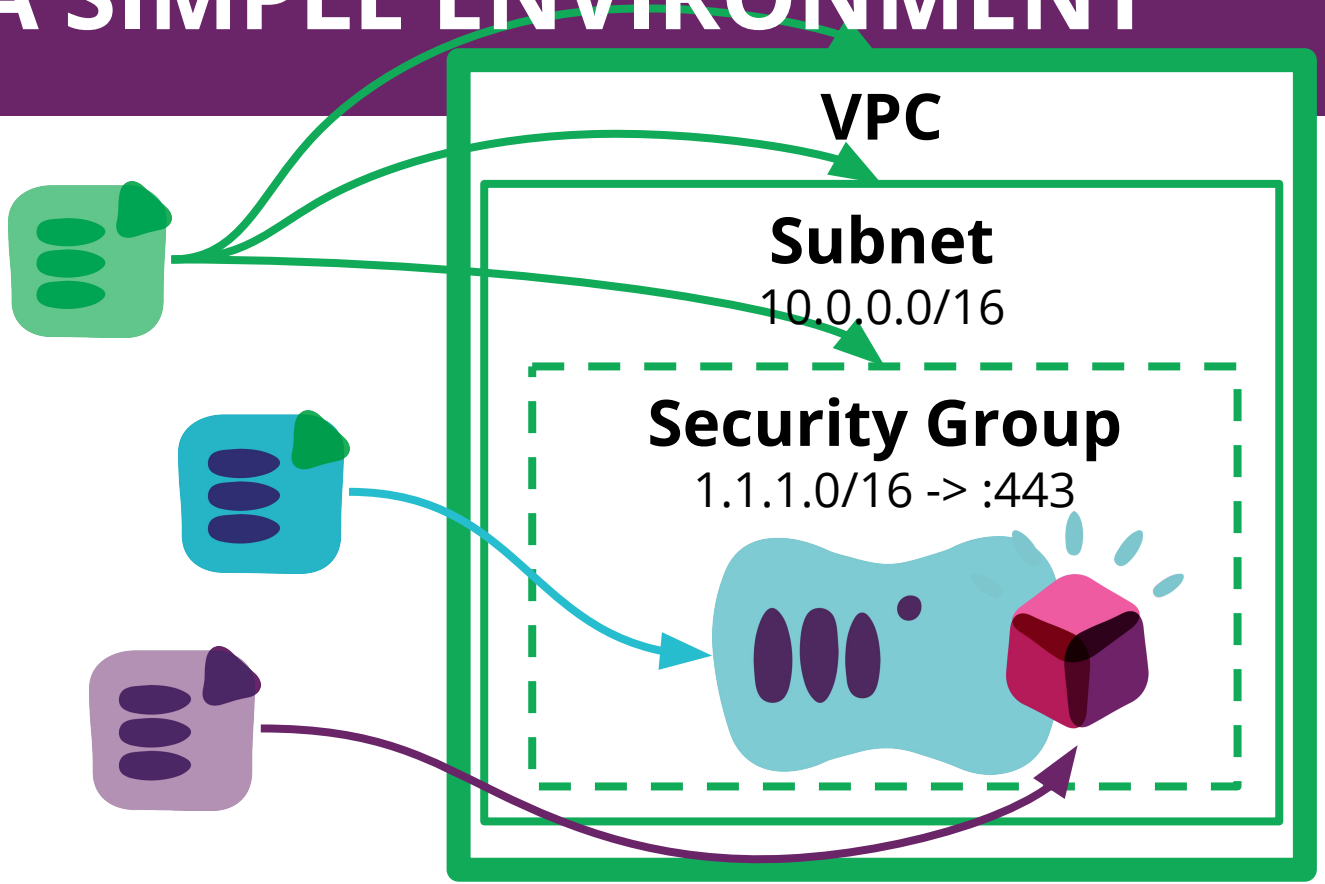
An example with a fairly simple environment

DEFINING A SIMPLE ENVIRONMENT

TERRAFORM FILE
Environment structure

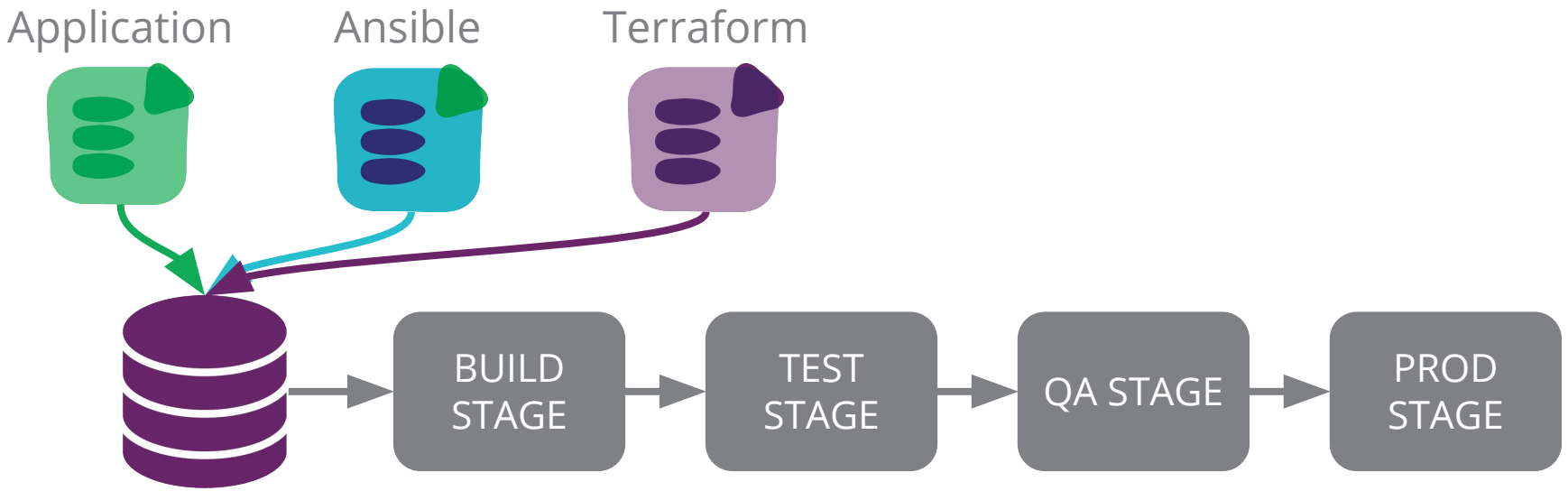
ANSIBLE PLAYBOOK
Server configuration

APPLICATION SOURCE
Deployable application



SIMPLE PIPELINE DESIGN

Deploy application, configuration, and infrastructure



SCALING

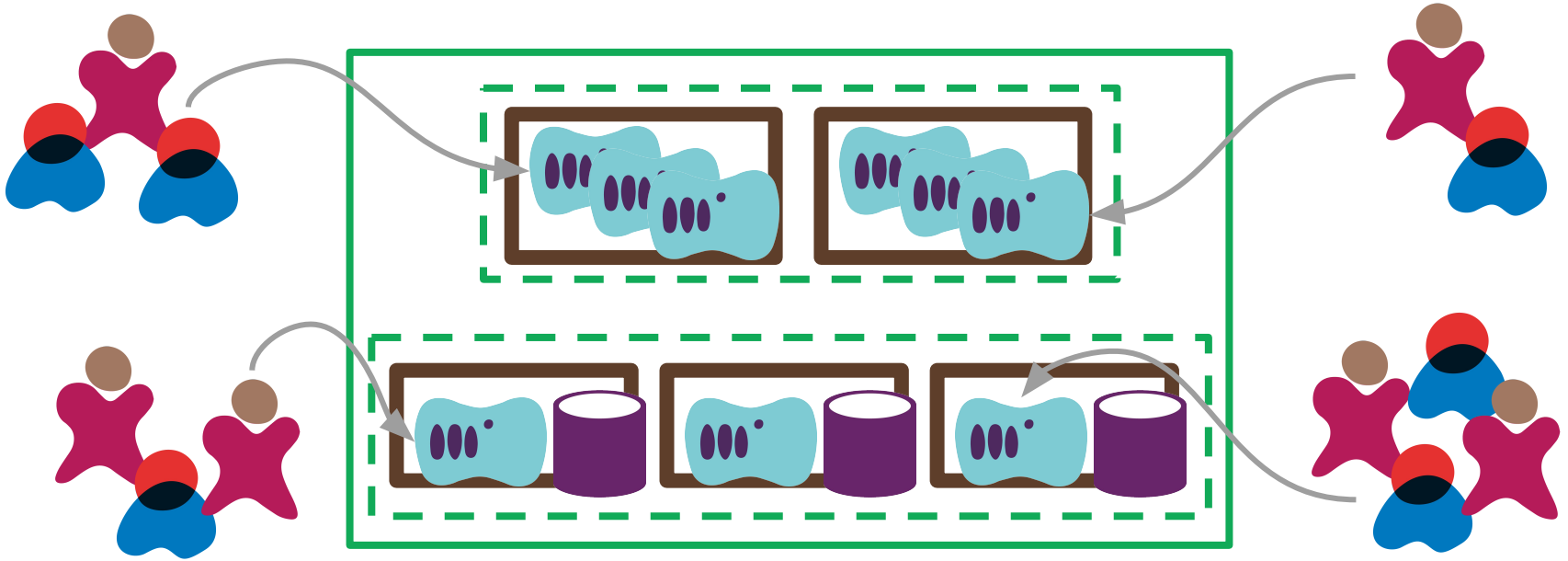
Handling more complex infrastructure

ALIGN INFRASTRUCTURE DESIGN TO TEAMS

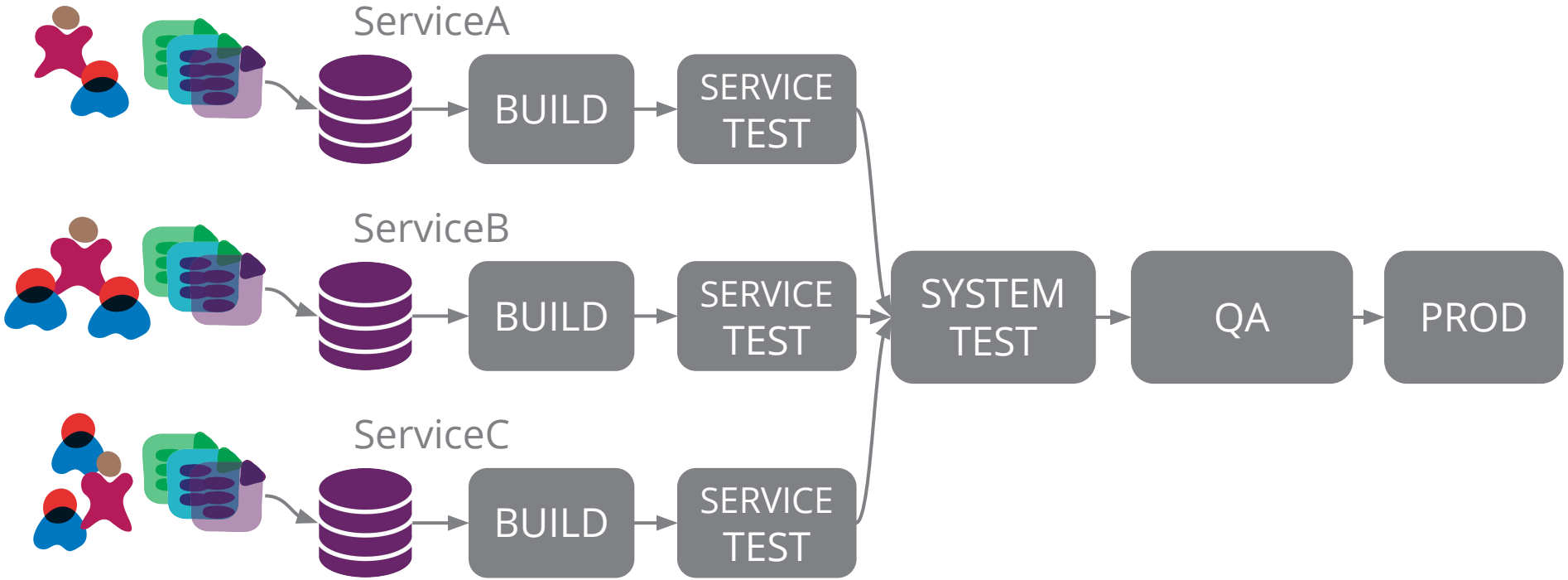
Ensure teams can make the changes they need **easily** and **safely**

COMPLEX ENVIRONMENTS

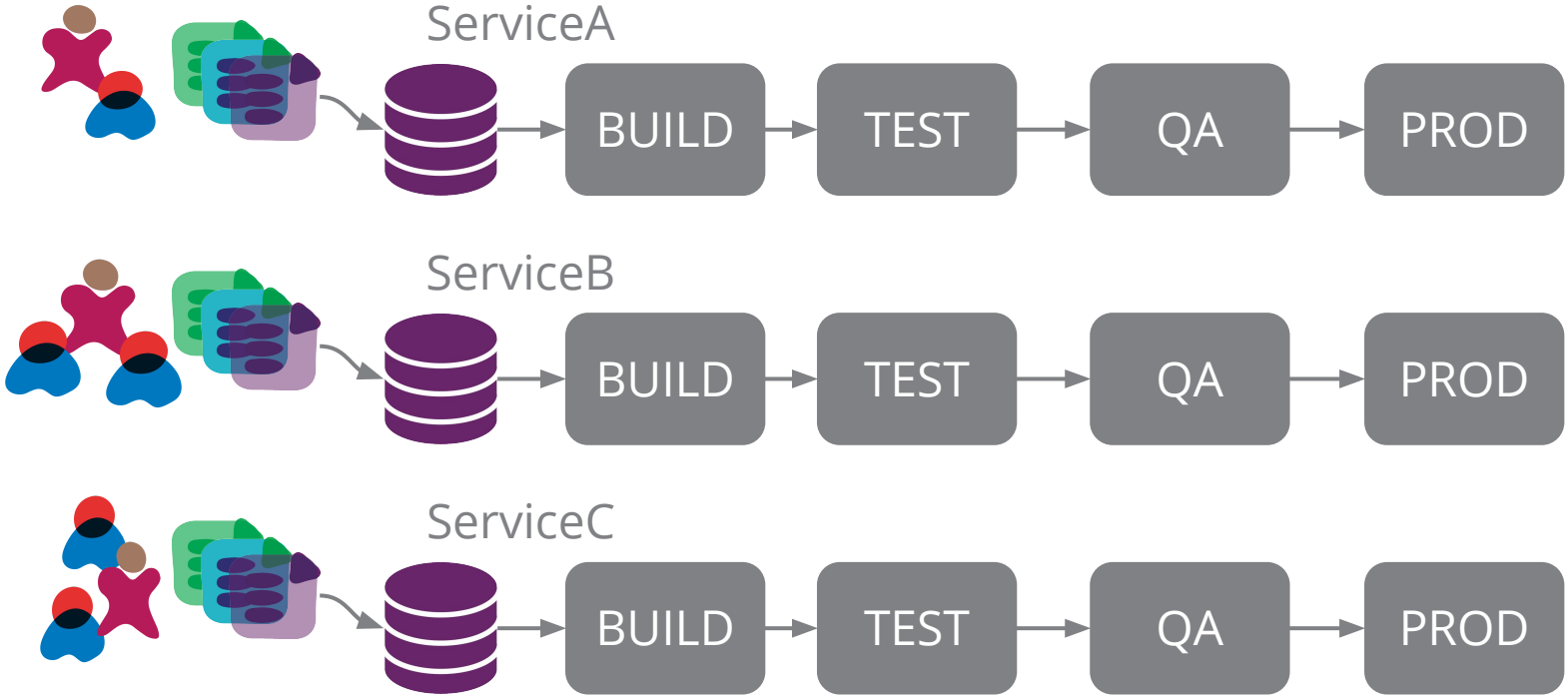
Infrastructure involving multiple teams



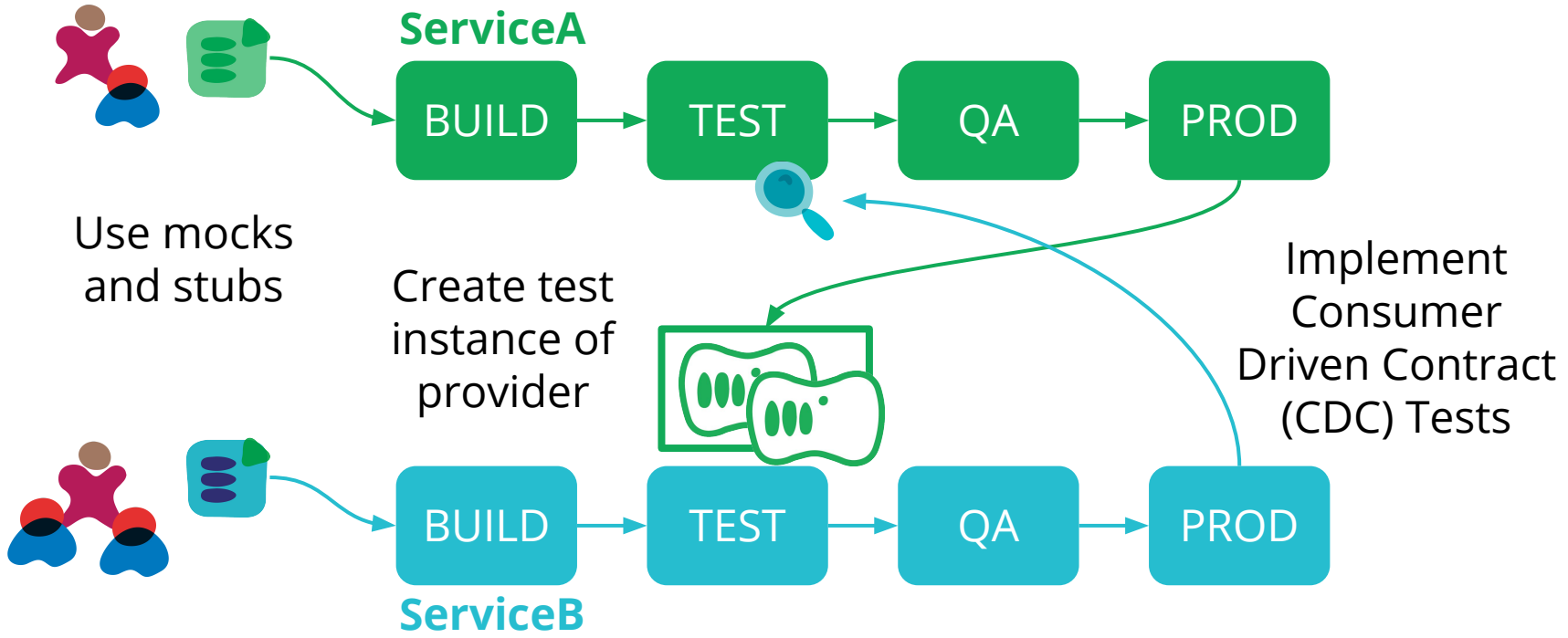
FAN-IN PIPELINE



DECOUPLED PIPELINES

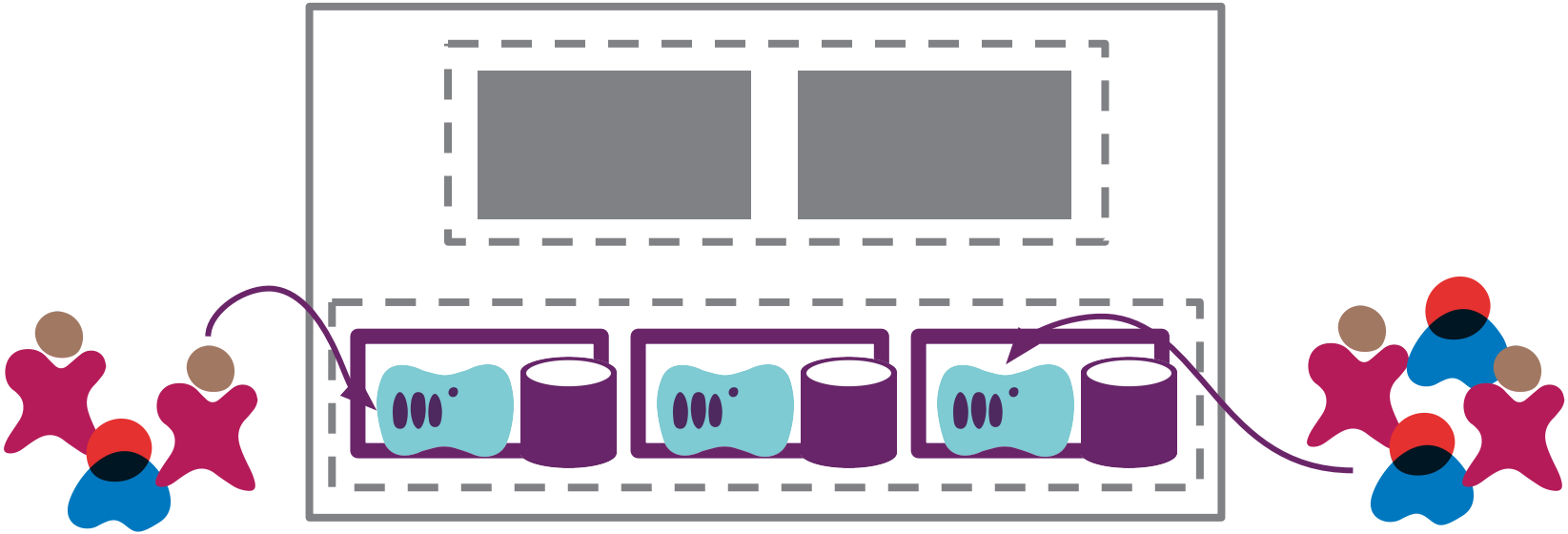


DEPENDENCIES



ISSUE: DUPLICATION

Multiple teams using similar systems, e.g. database clusters



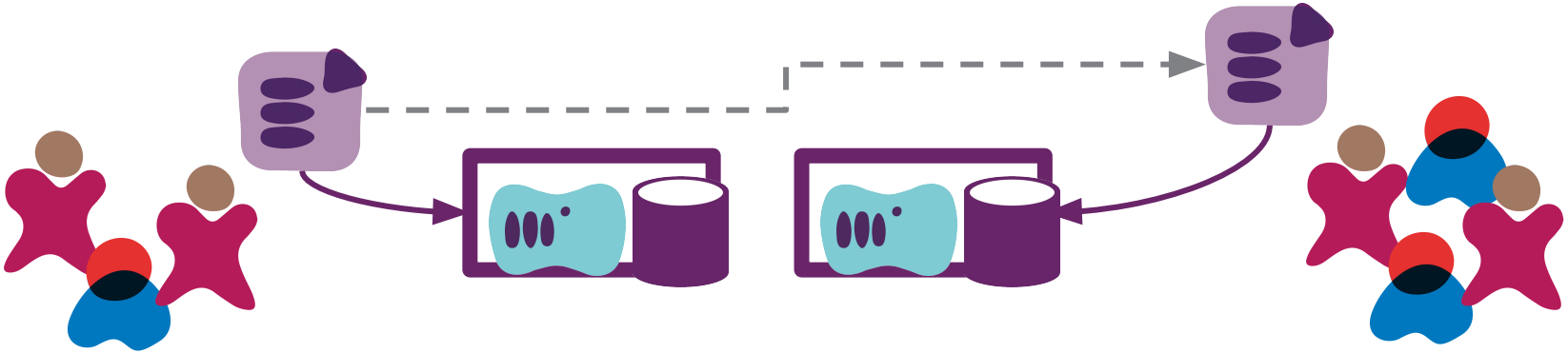
RE-USE BY FORKING DEFINITIONS

Disadvantages:

- Divergence and Inconsistency

Advantages:

- Avoid tight coupling
- Handles diverse requirements



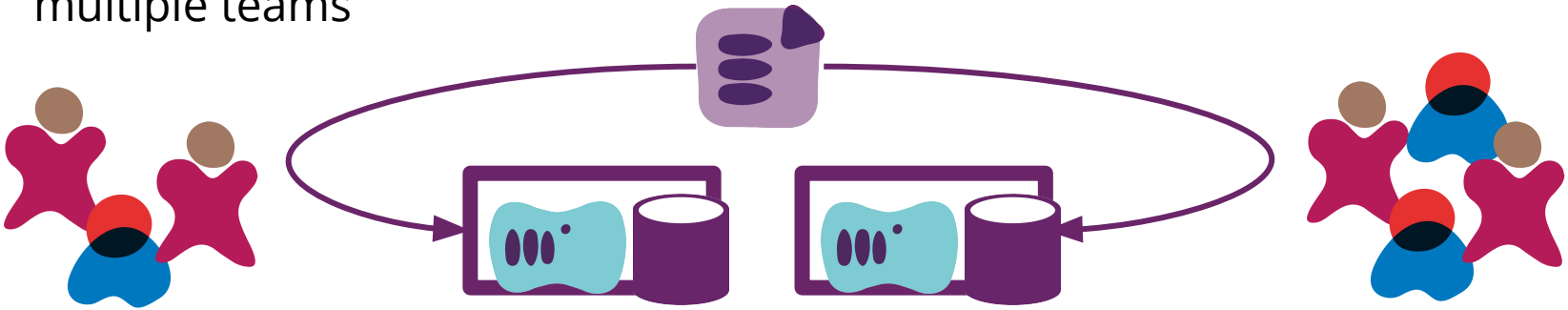
RE-USE WITH DEFINITION LIBRARIES

Challenges:

- Avoid tight coupling, so teams aren't blocked when making changes
- Ownership of code shared by multiple teams

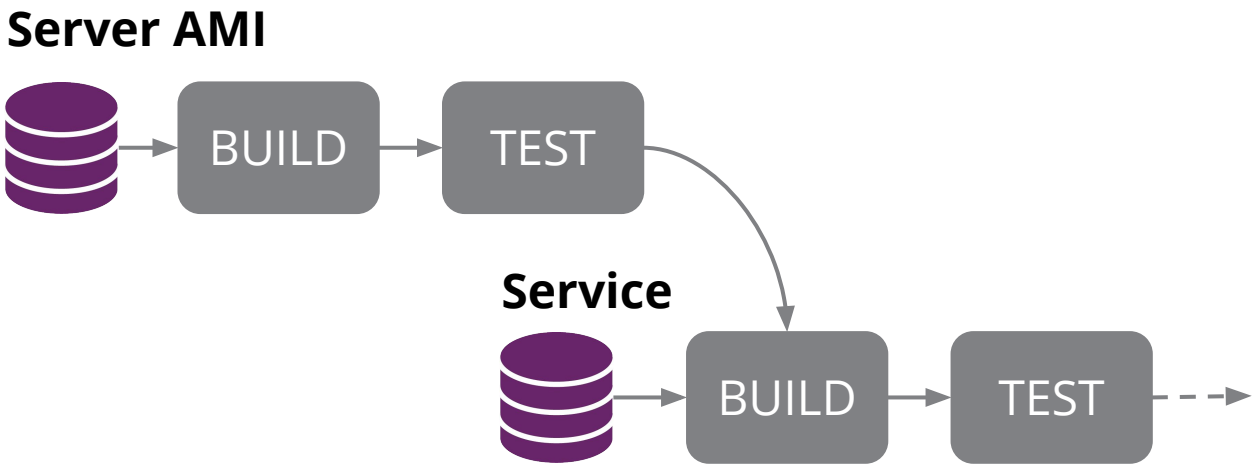
Guidance:

- Use separate pipelines for each
- Use CDC & other dependency testing strategies

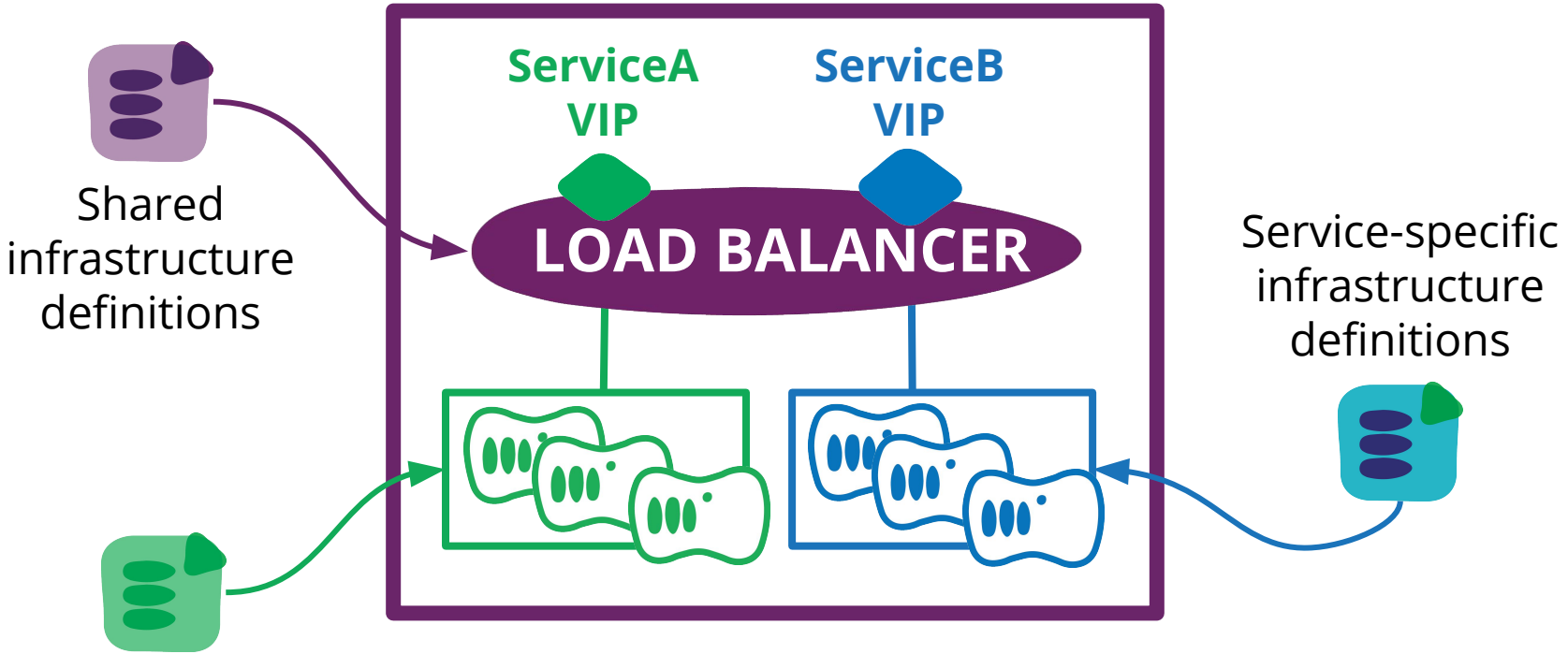


LIBRARY PIPELINE

Test shared definitions before pulling them into dependent pipelines

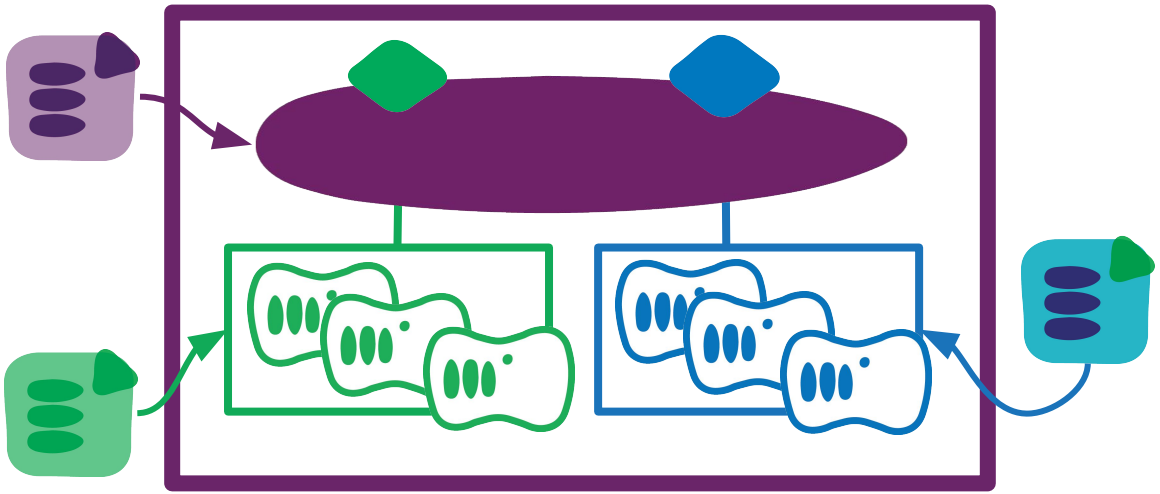


ISSUE: SHARED ELEMENTS



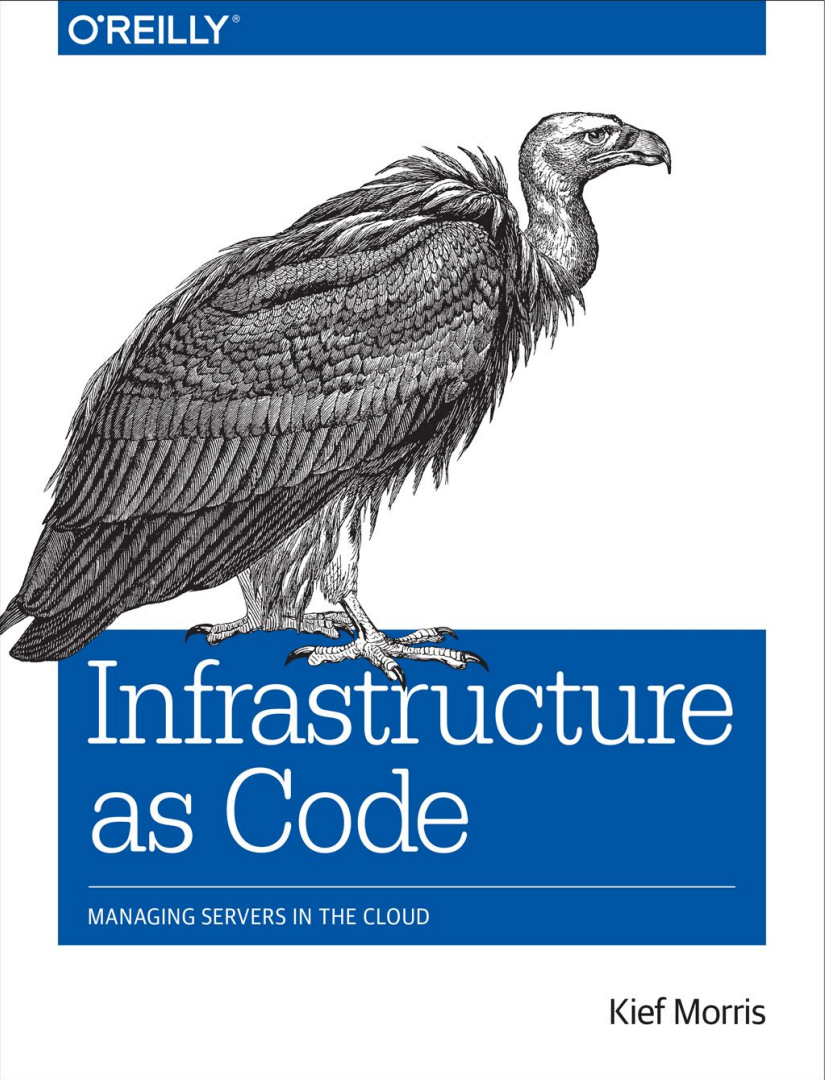
SHARING ELEMENTS

Avoid monoliths - optimize to simplify making changes



OUTCOMES

- Quickly **provision** and **evolve** infrastructure
- Effortlessly roll out **fixes**
- Keep systems **consistent** and up to date
- Spend time on **high value** work



kief@thoughtworks.com

Cloud Practice Lead (UK)

DevOps, Continuous Delivery, Agile Ops

ThoughtWorks®

Book: http://oreil.ly/1JKIBVe
Site: http://infrastructure-as-code.com
Twitter: @kief

QCon NEW YORK