Practical approach to Infrastructure as Code - how to effectively program your cloud

Tomasz Cholewa

OSEC Forum 2017

whoami

Automation freak

DevOps Enthusiast

Cloud Infrastructure Architect (AWS)

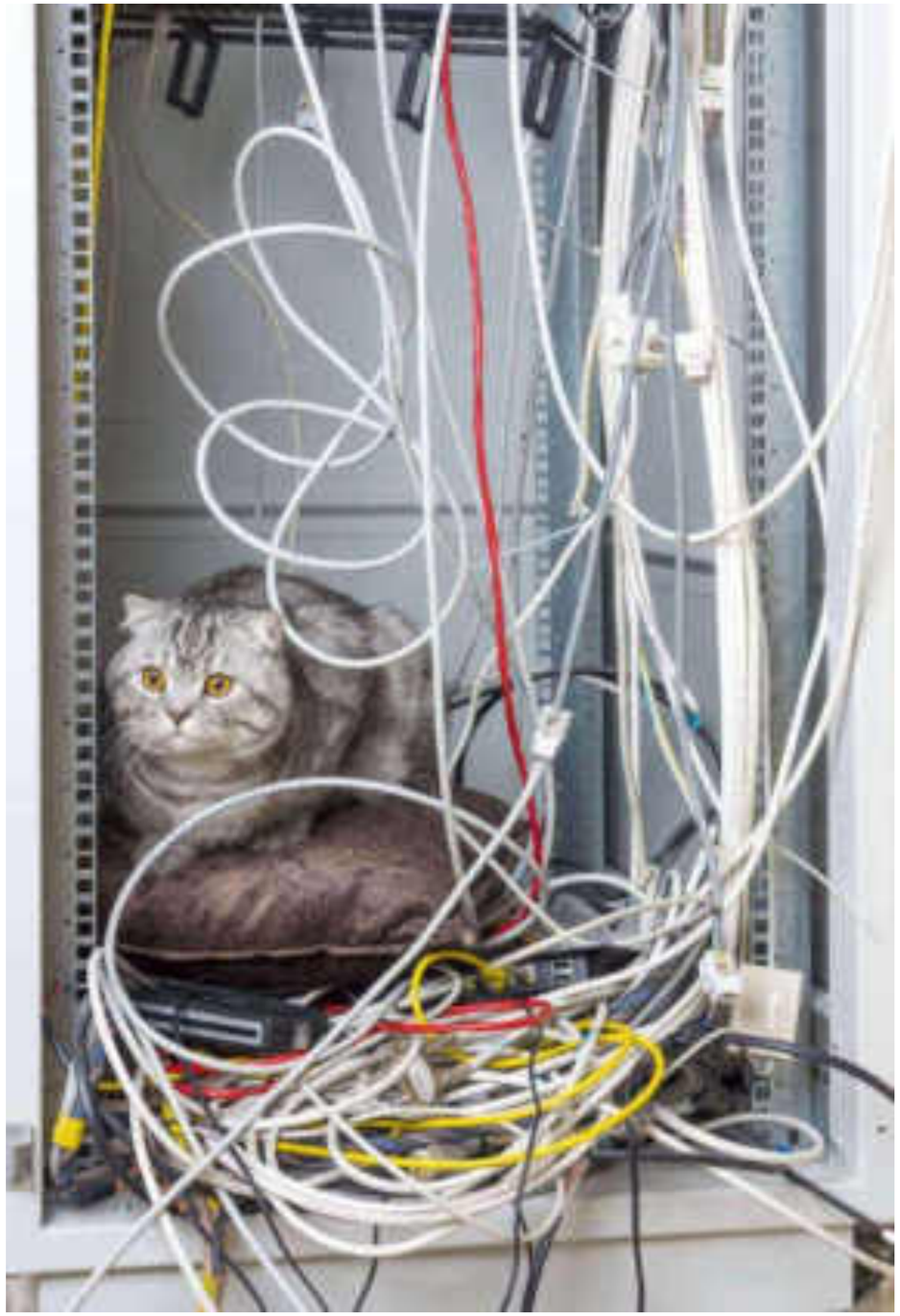
Infrastructure as Code practitioner

Certificate collector

redhat.
CERTIFIED
ARCHITECT


Expectations

Reality



Why do we need IaC?

Progress & Innovation

Scaling

Cloud & DevOps

Money

Definition

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

CloudFormation = IaC

CloudFormation ≠ IaC

CloudFormation < IaC

Rule 1

IaC is not about tools, but a process

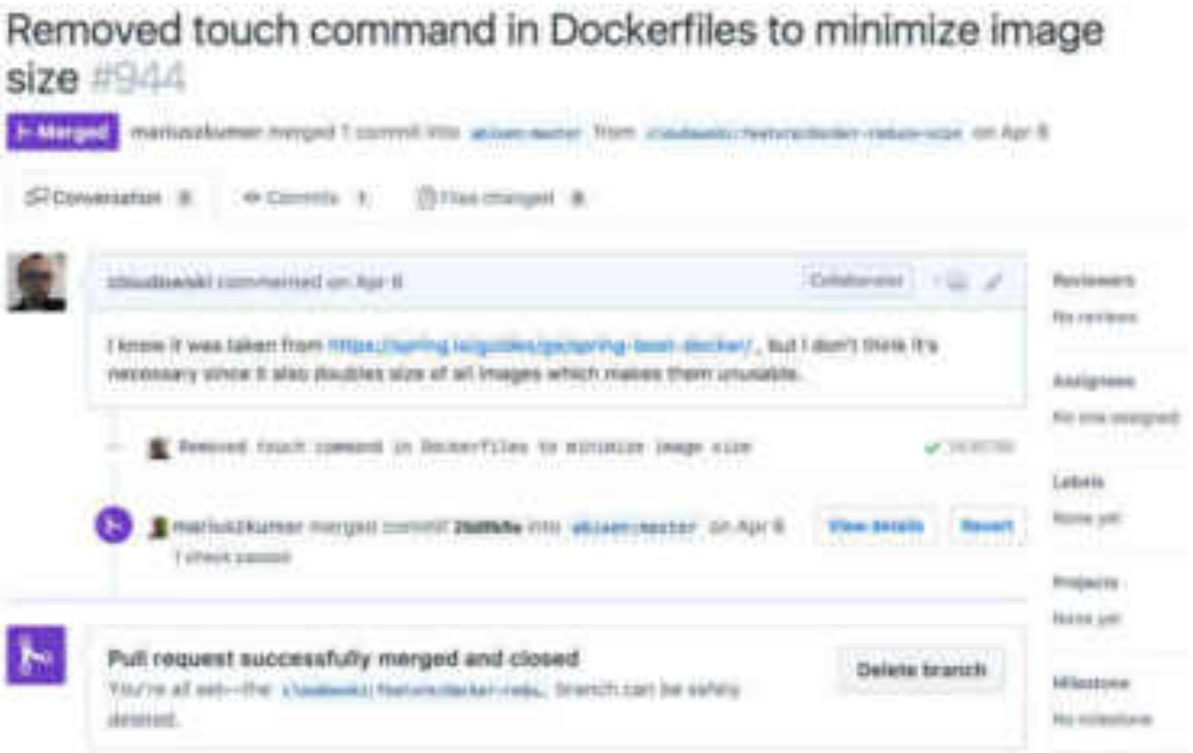
Change control

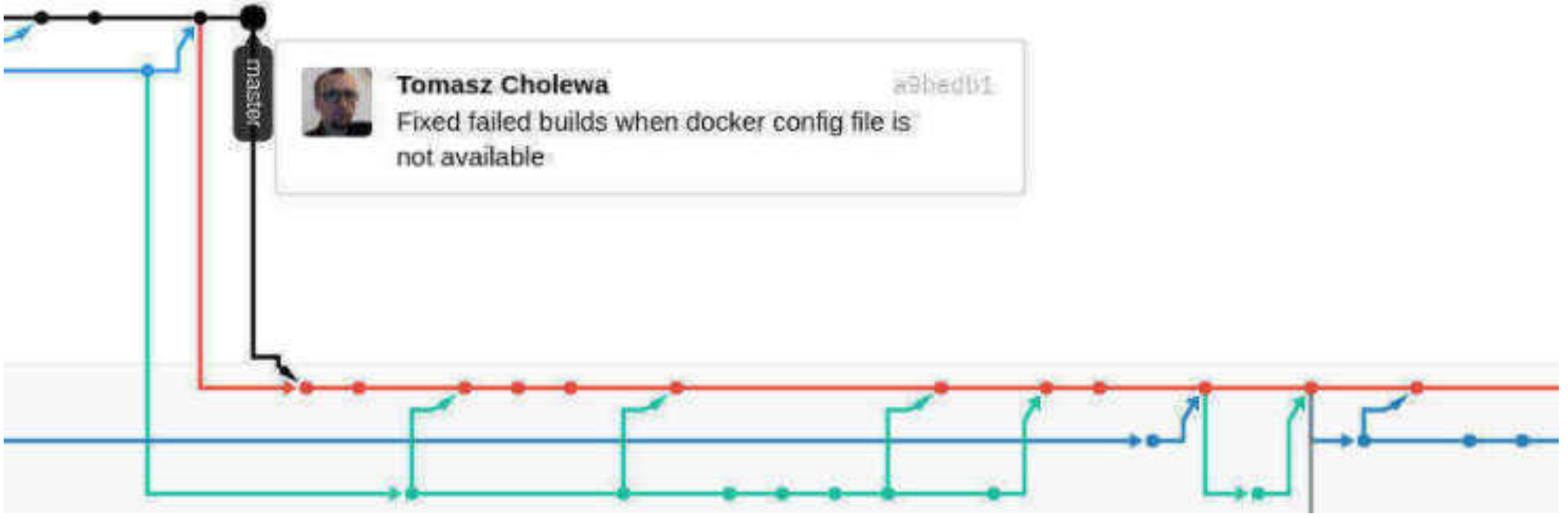

VERSION
AUTOMATE
ALL THE THINGS
memegenerator.net

GIT

Code management practices

- Feature branches
- Code review
- Pull requests
- Tagging




master
Tomasz Cholewa
Fixed failed builds when docker config file is not available

```
commit c73a1333b5d31cdbb0ccab376c4dfb277e7c1948
Author: Tomasz Cholewa <tomasz@cloudowski.com>
Date:   Wed Mar 1 21:03:53 2017 +0100

    Added gc stats to grafana dashboard. Fixed #754

commit 0ad02b54d895963c68cb8ccdf5373ddd7ffe7c48
Author: Tomasz Cholewa <tomasz@cloudowski.com>
Date:   Wed Mar 1 21:15:19 2017 +0100

    Replaced host volumes with named volumes for database. Fixed #766

commit 2b8a0988d19b73b8f8be9255b63a3eff3b2a9aa9
Author: Tomasz Cholewa <tomasz@cloudowski.com>
Date:   Tue Feb 28 17:58:04 2017 +0100

    Change name for load balancer in docker-compose for web-client

commit f9872f34c15d127864a223443f4884fa68364440
Author: Tomasz Cholewa <tomasz@cloudowski.com>
Date:   Mon Feb 27 21:24:05 2017 +0100

    Added screenshots for wiki monitoring and logging sections

commit abc583f4a35af0b5ea3938e0119237147d413b88
Author: Tomasz Cholewa <tomasz@cloudowski.com>
Date:   Mon Feb 27 20:48:34 2017 +0100

    Added postgres input in telegraf and added core. Resolved #741

commit b2991e8d313b5e3a3931202c7eb1a7a1747d1734
```

Source: https://github.com/abixen/abixen-platform

Repository content

Yaml,JSON,HCL

Dockerfile

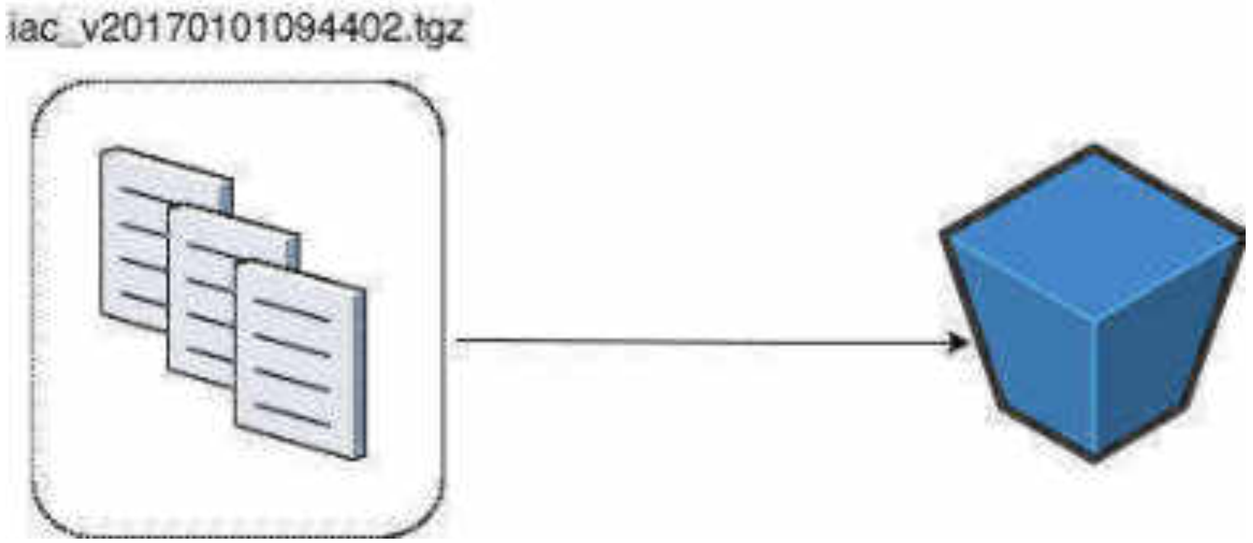
Scripts (shell, python, ruby, …)

Tests

Configuration

Misc

Artifacts



Versioning of external components

Python modules (including ansible) - requirements.txt

Ruby Gems - Gemfile

Custom wrappers for external tools (example https://github.com/cloudowski/terraform-wrapper)

Docker containers (never use “latest” tag)

```
boto
ansible==2.2.1
ansible-lint==3.4.11
pep8
```

requirements.txt

```
source 'https://rubygems.org'
gem "test-kitchen", "1.8.0"
gem "kitchen-vagrant"
gem "serverspec"
gem "kitchen-docker"
gem "kitchen-ansible"
gem "kitchen-verifier-serverspec"
gem "kitchen-ec2"
#gem "ec2"
gem "aws-sdk", '>2.4.0'
```

Gemfile

Rule 2

You can’t do IaC without versioning

Consistency

It works for me

=

it works on my environment

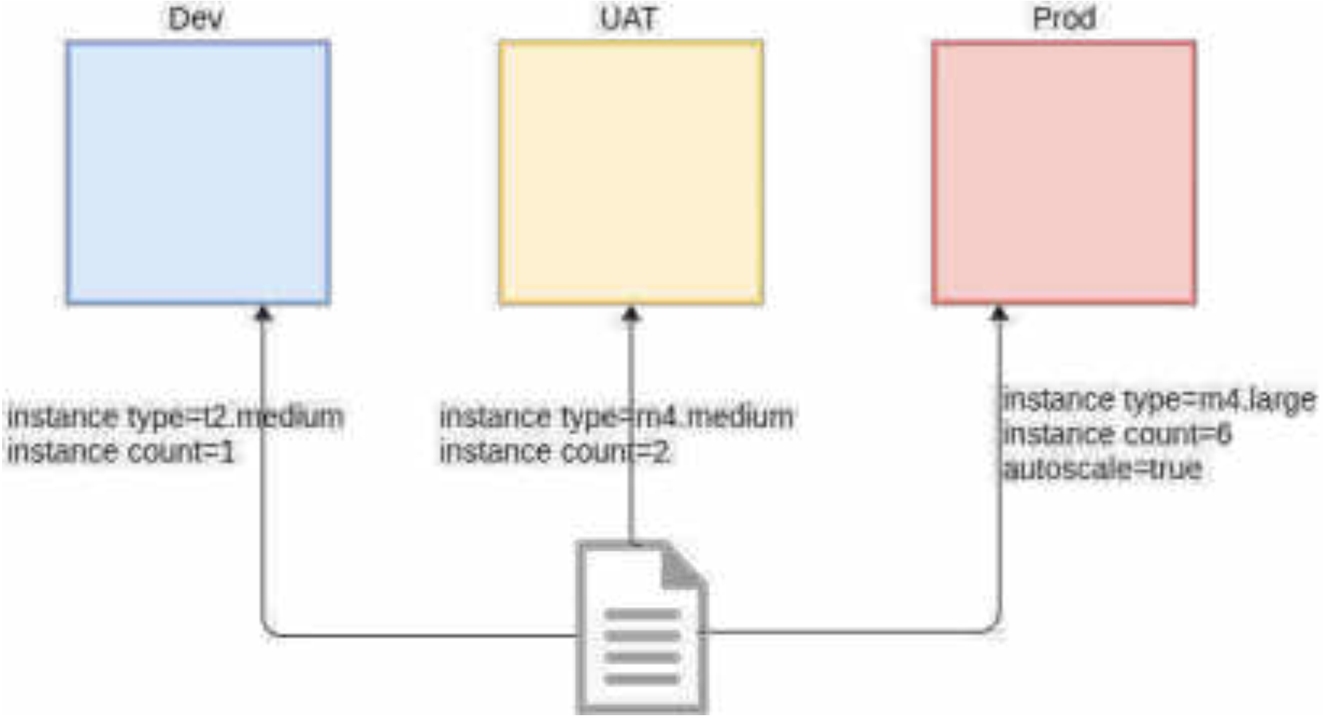
Consistency

Greater autonomy - self-serviced environments

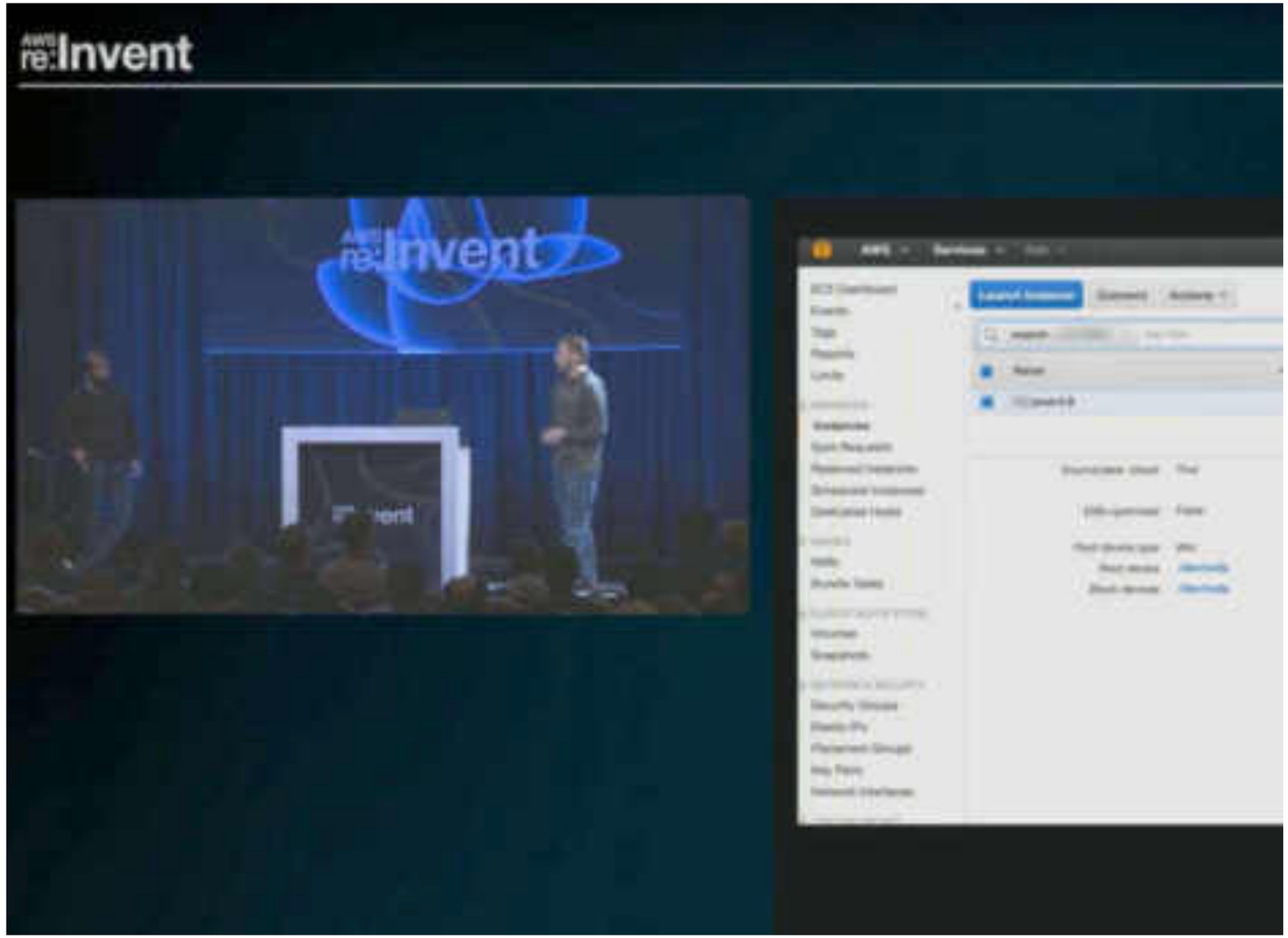
AWS Service Catalog

Docker, Vagrant based environments

AWS OpsWorks, Elastic Beanstalk

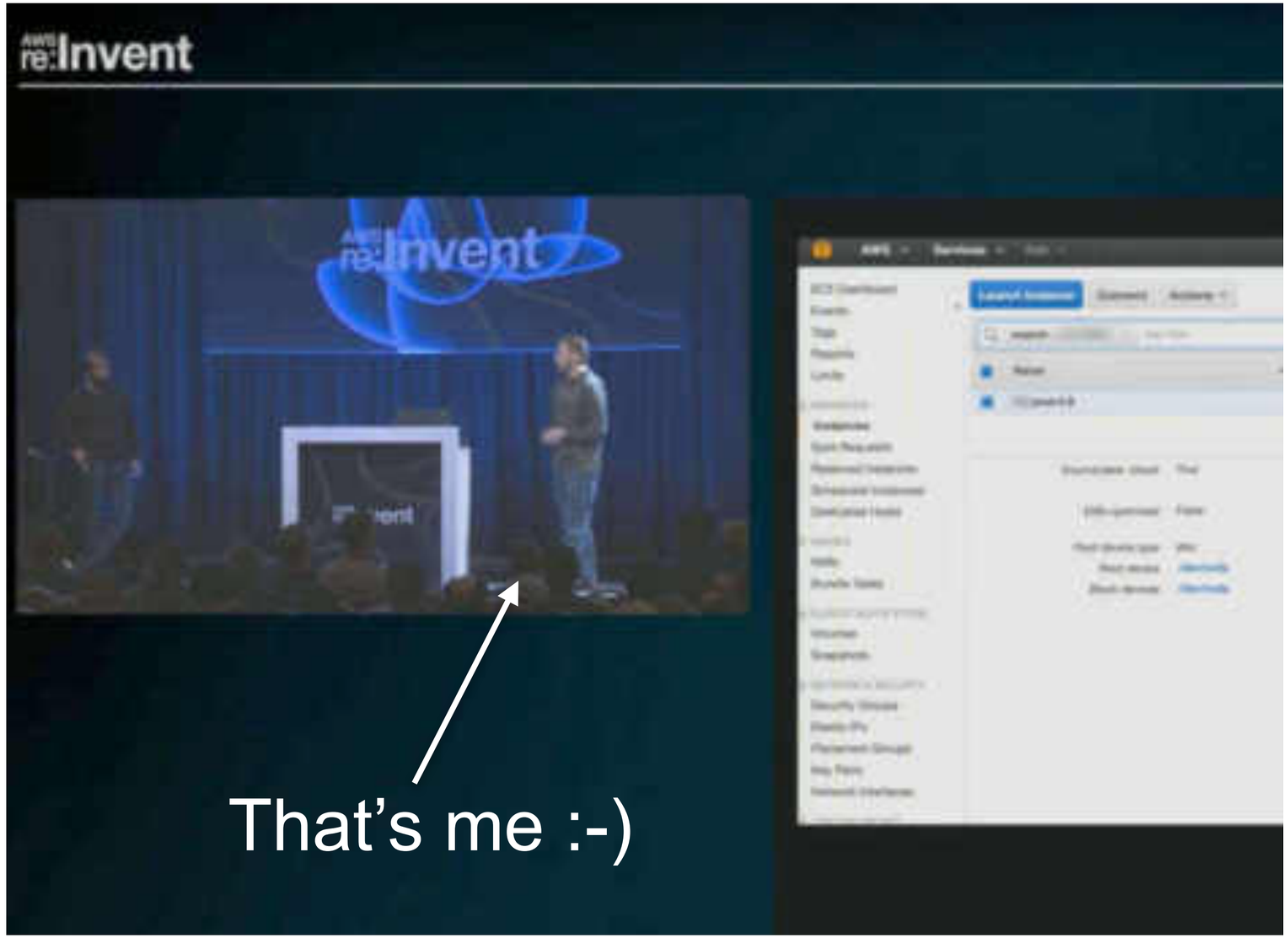


Immutable Infrastructure



[AWS re:Invent 2016: Life Without SSH: Immutable Infrastructure in Production (SAC318)]

Immutable Infrastructure



[AWS re:Invent 2016: Life Without SSH: Immutable Infrastructure in Production (SAC318)]

Configuration management

Configuration per environment

Ansible inventories - one per environment

Dynamic inventories per “Env” AWS tag

- ec2.py

- Consul

Dedicated repository with proper policy

Secret and sensible data management

Ansible Vault

HashiCorp Vault

NEVER, EVER in plaintext in repo!

No credentials

- IAM Instance Profiles

- IAM Roles for ECS Task

Recipe for new environment

Tools + Code + Config = Environment

CloudFormation	CFN Templates	Variables
Terraform	Terraform HCL	Secret/Sensitive Data
Ansible	Playbooks, Roles	Extra code

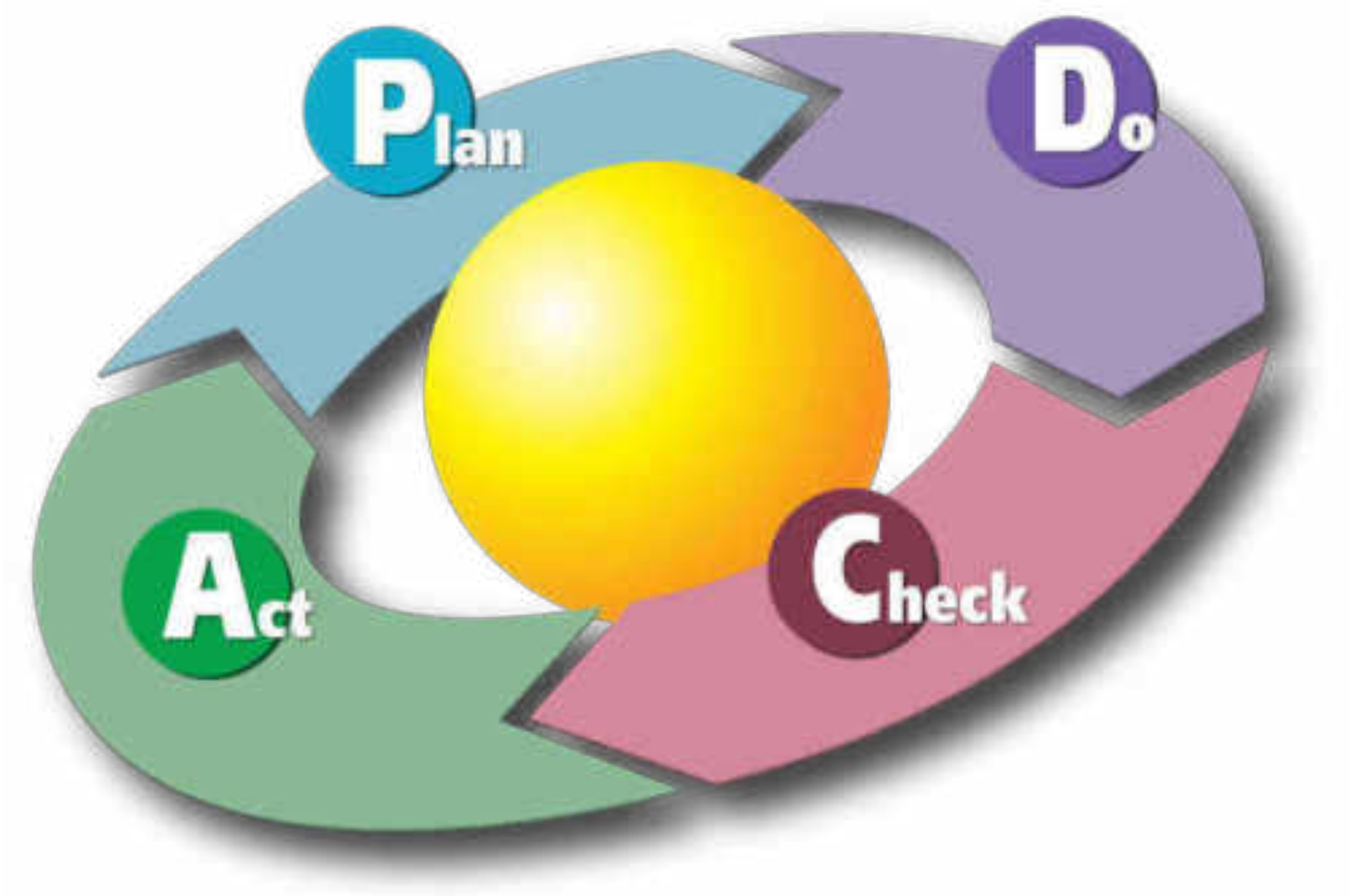
Rule 3

You can’t name it IaC if you can’t create new env quickly

Antifragile Infrastructure

Apparently S3 is not Antifragile (yet?)

Continuous Improvement



Multicloud requires IaC

Rule 4

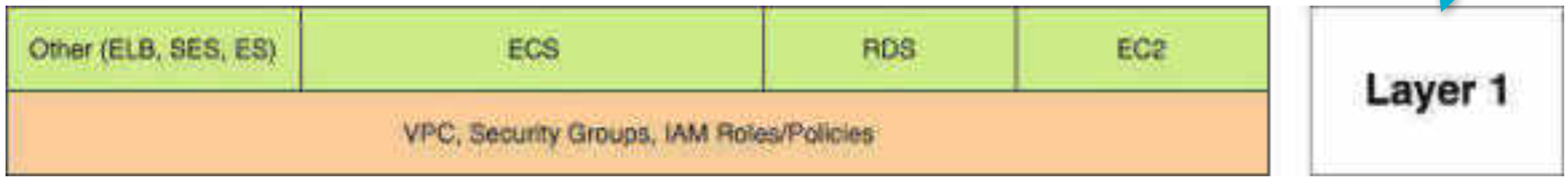
You can’t do IaC without continuous improvement of the process

Provisioning

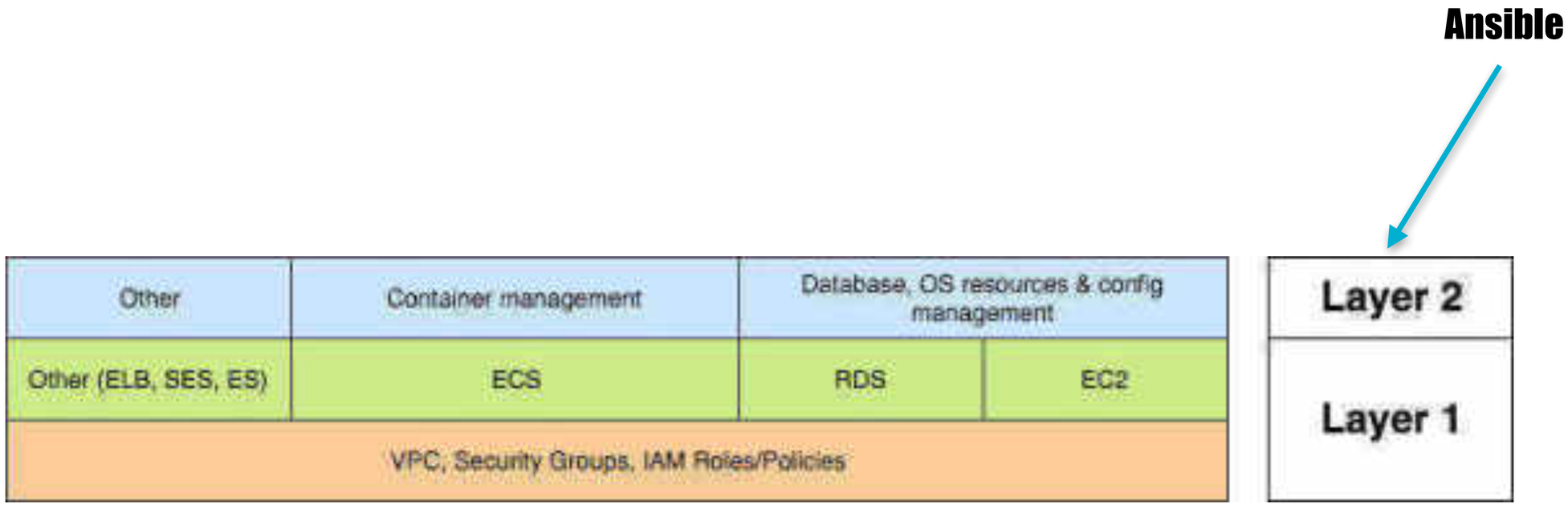
Provisioning layers

Provisioning layers

CloudFormation Templates
Terraform Modules



Provisioning layers



Tagging

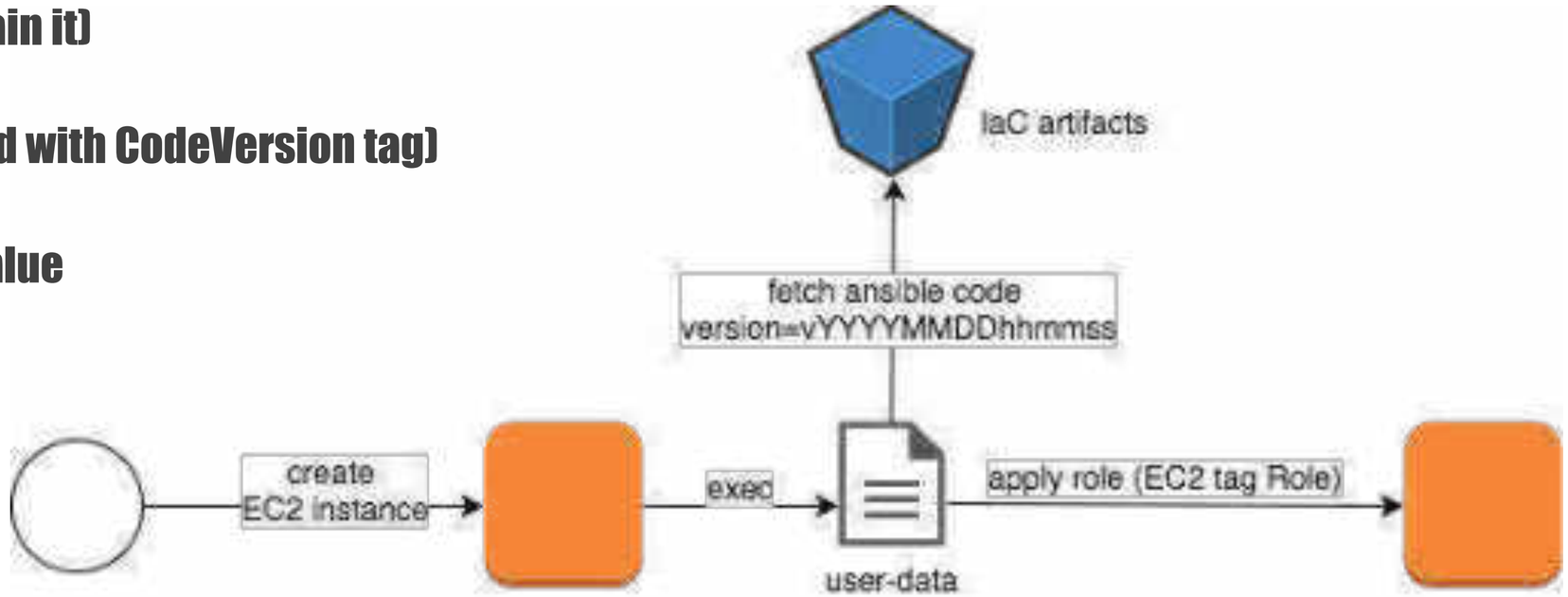
Tagging as a part of deployment process

Recommended IaC related tags:

- Env - environment name (consistent with naming in configuration part of the code)
- CodeVersion - version string of the code which provisioned resource
- Role - for EC2 instances it may be used for startup provisioning (e.g. Ansible)

Ansible provisioning

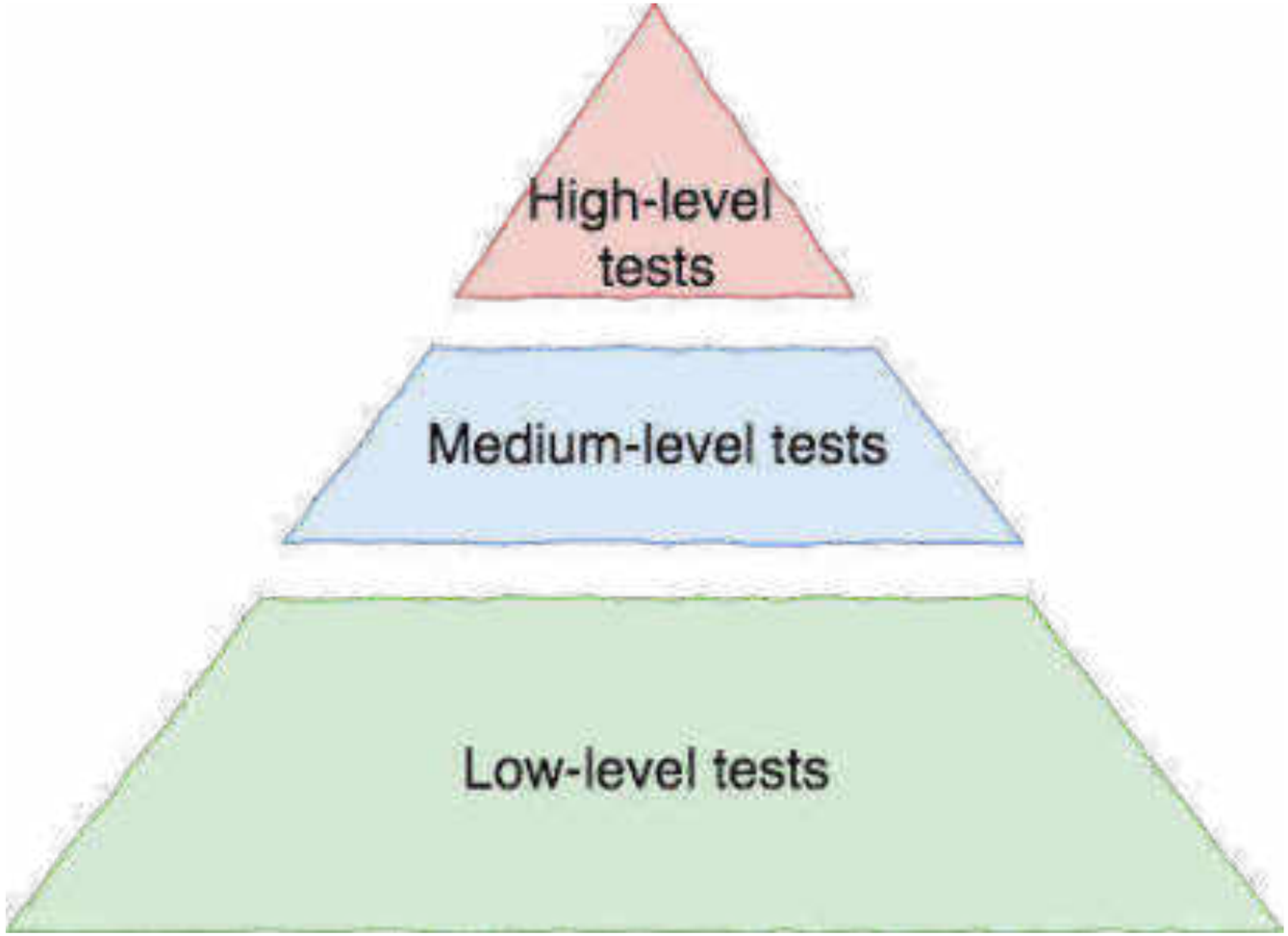
1. Launch instance
2. Exec UserData code
    1. Install ansible (if AMI doesn't contain it)
    2. Fetch ansible code (version aligned with CodeVersion tag)
3. Apply role(s) based on Role EC2 tag value



Provisioning layers

(Software) Continuous Deployment Processes

Testing



Pyramid of tests

Low level

Check definition files (syntax, style)

Linters

- Pep8
- ansible-lint
- CloudFormation - validate-template command

Configuration checks

Launched automatically by build server (on PR)

Medium level tests

Test-kitchen (http://kitchen.ci)

Provision test resources (locally)

Regression testing

Compliance testing

```
---
driver:
  name: vagrant

provisioner:
  name: ansible_playbook
  roles_path: roles
  hosts: tomcat-servers
  require_ansible_repo: true
  ansible_verbose: true
  ansible_version: latest
  require_chef_for_busser: false
  additional_ssh_private_keys:
  - /mykey/id_rsa

platforms:
  - name: nocm_centos-6.5
    driver_plugin: vagrant
    driver_config:
      box: nocm_centos-6.5
      box_url: http://puppet-vagrant-boxes.puppetlabs.com/centos-65-x64-virtualbox-nocm.box
      network:
      - ["forwarded_port", {guest: 8080, host: 8080}]
      - ["private_network", {ip: '192.168.33.11'}]
```

Medium level tests

Test-kitchen verifiers

- ServerSpec
- InSpec

```
require 'spec_helper'

describe package('httpd'), :if => os[:family] == 'redhat' do
  it { should be_installed }
end

describe package('apache2'), :if => os[:family] == 'ubuntu' do
  it { should be_installed }
end

describe service('httpd'), :if => os[:family] == 'redhat' do
  it { should be_enabled }
  it { should be_running }
end

describe service('apache2'), :if => os[:family] == 'ubuntu' do
  it { should be_enabled }
  it { should be_running }
end

describe service('org.apache.httpd'), :if => os[:family] == 'darwin' do
  it { should be_enabled }
  it { should be_running }
end

describe port(80) do
  it { should be_listening }
end
```

High-level tests

Launched on dedicated environment and/or live environments

Security related (e.g. SSL, ingress/egress traffic, CVE vulnerabilities)

Performance & load testing (e.g. AutoScaling responsiveness)

“Temporary”/non-compliant resources discovery

Test like a Pro!

Simian Army

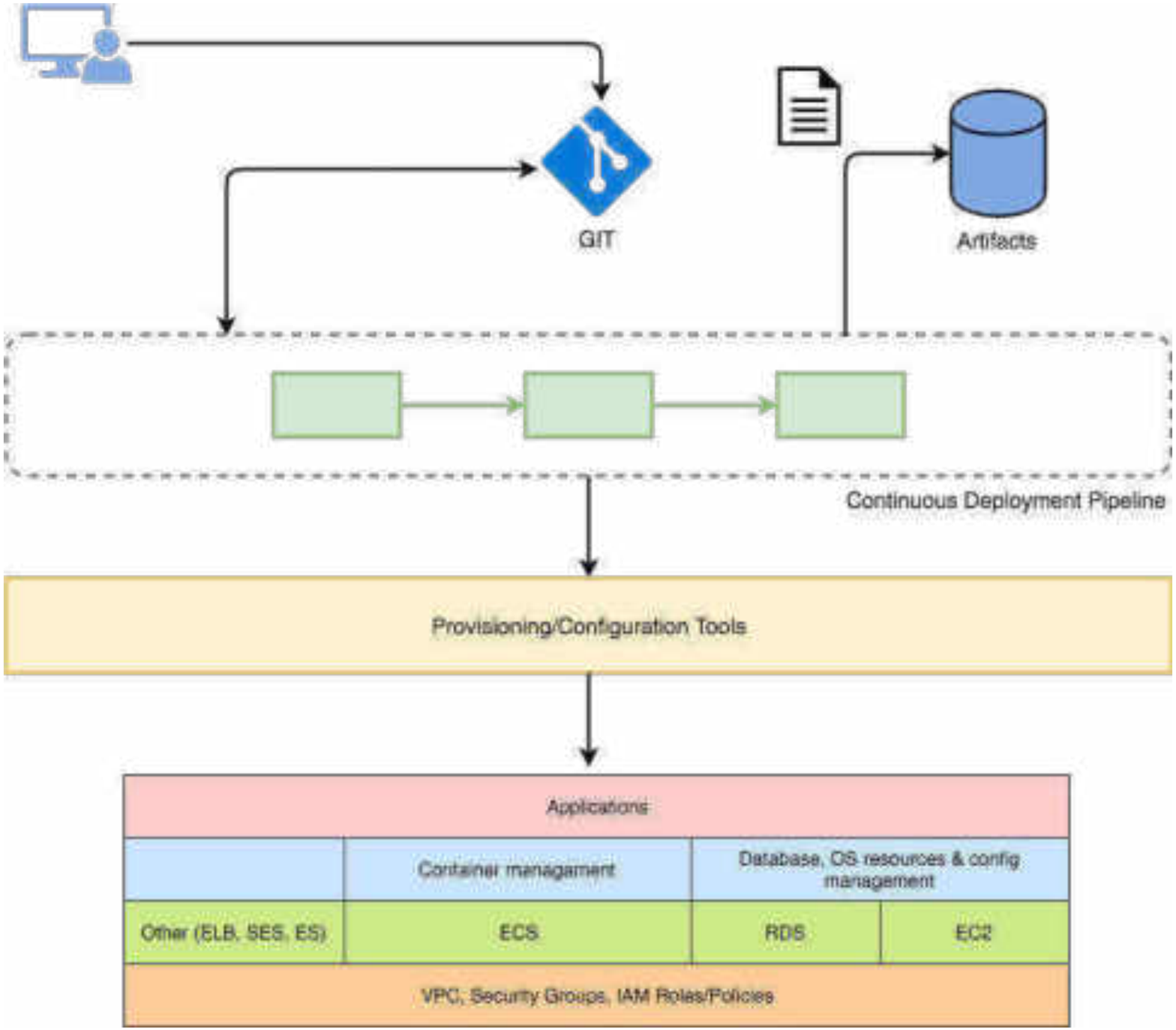
- Chaos Monkey
- Chaos Gorilla
- Chaos Kong
- Conformity Monkey
- Doctor Monkey
- Latency Monkey
- Janitor Monkey
- Security Monkey

Rule 5

You can’t do IaC without proper testing

Continuous Deployment Pipeline

Overview


GIT
Artifacts
Continuous Deployment Pipeline
Provisioning/Configuration Tools
Applications
Container management
Database, OS resources & config management
Other (ELB, SES, ES)
ECS
RDS
EC2
VPC, Security Groups, IAM Roles/Policies

Steps

1 - send code to git repository

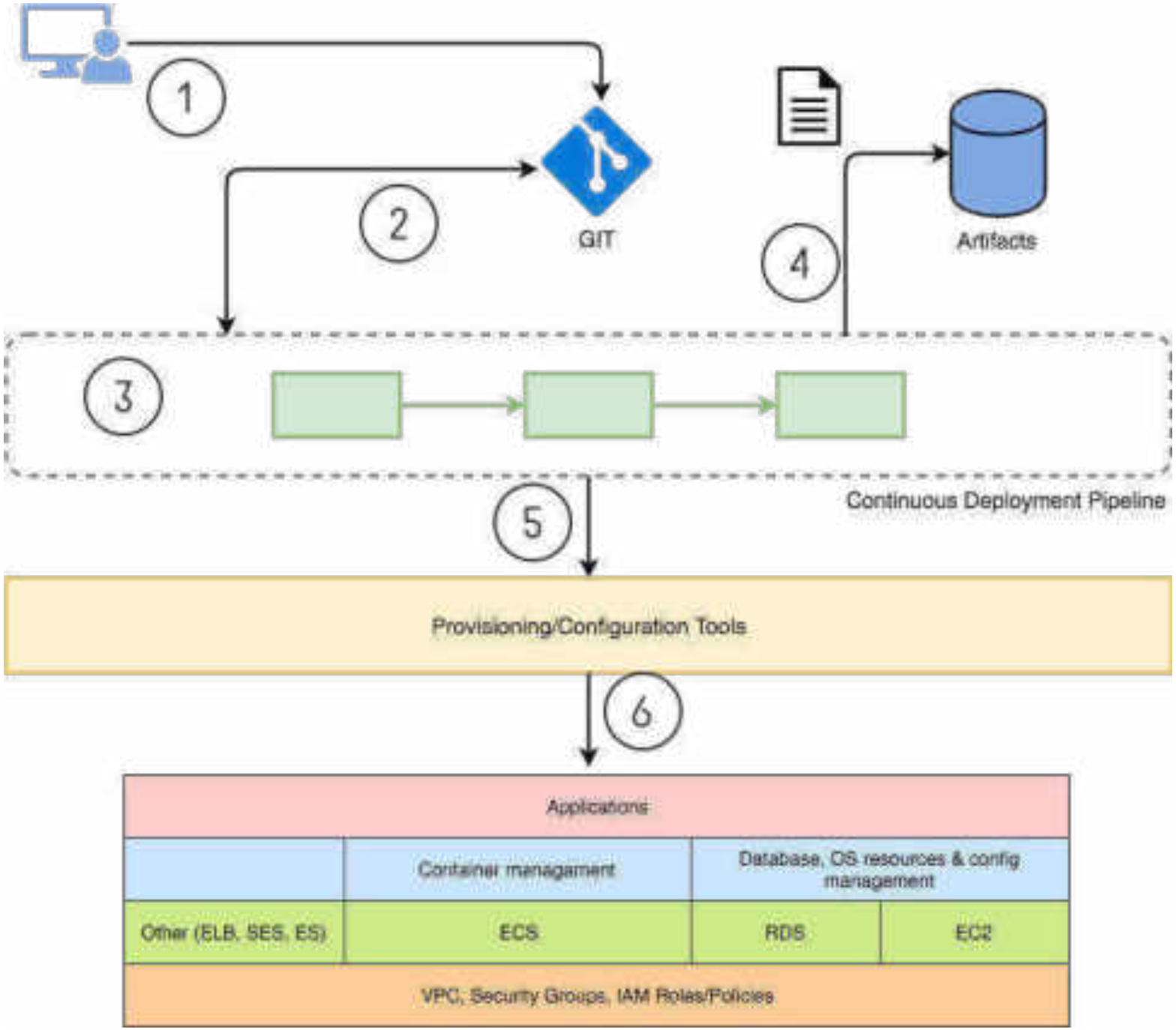
2 - initialize/start pipeline

3 - run pipeline stages

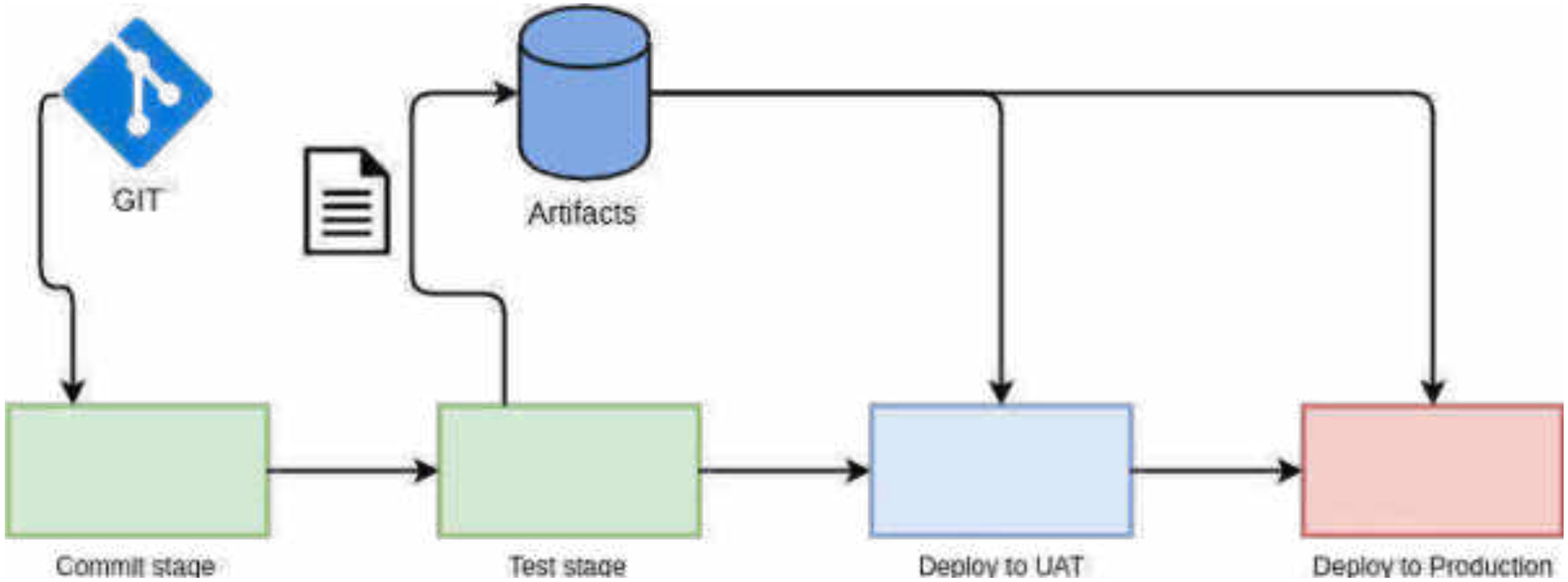
4 - publish artifacts

5 - deploy code on particular environment

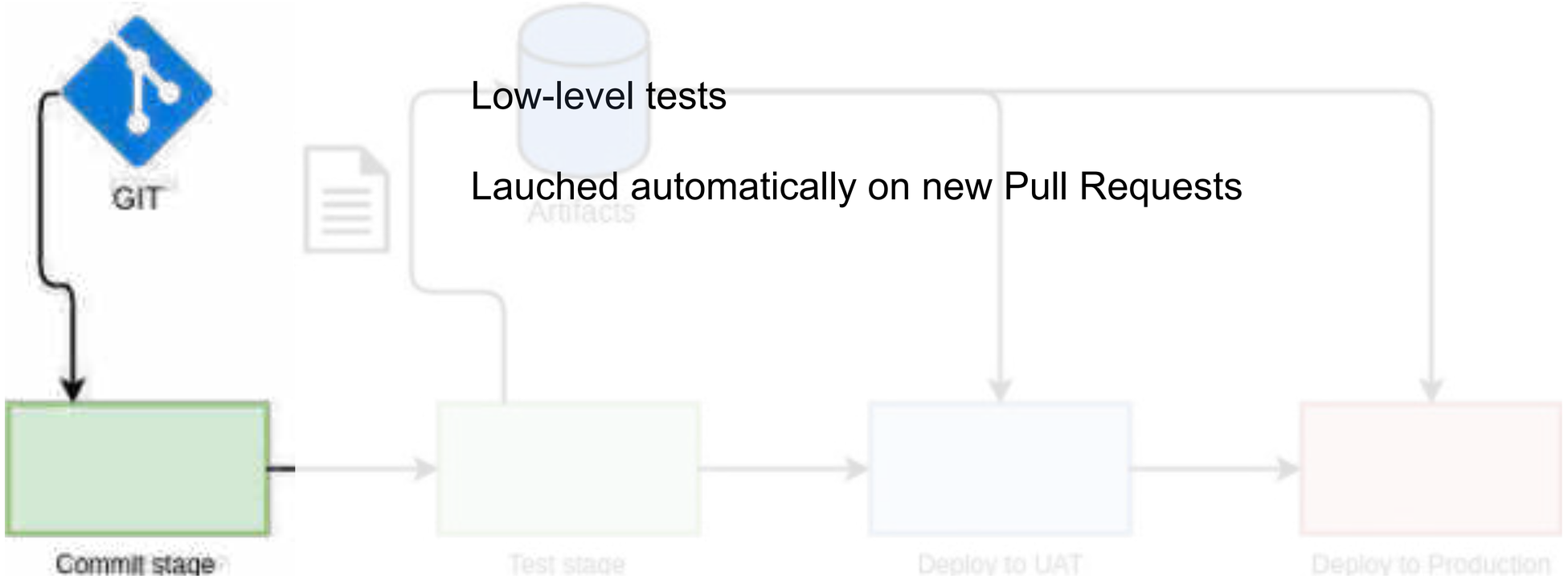
6 - provision environment layers



Pipeline


GIT
Artifacts
Commit stage
Test stage
Deploy to UAT
Deploy to Production

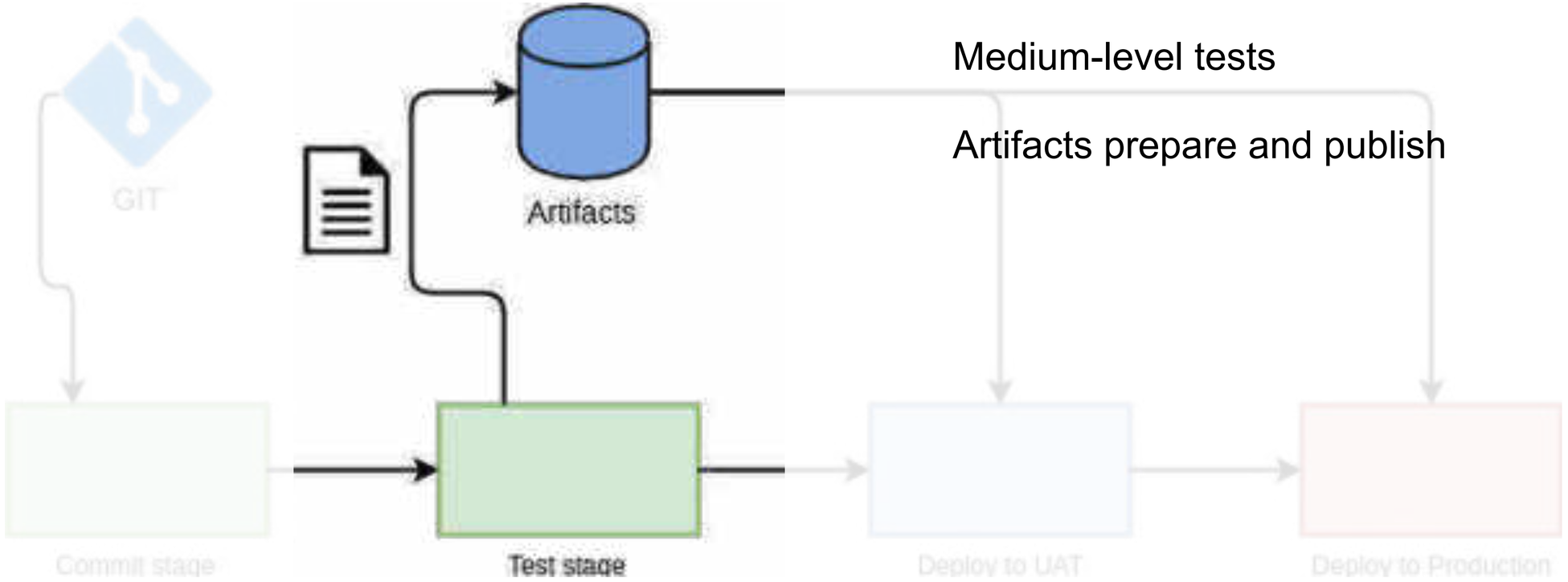
Commit stage



Low-level tests

Lauched automatically on new Pull Requests

Test stage



Medium-level tests

Artifacts prepare and publish

Deployment

Provision resources for environment

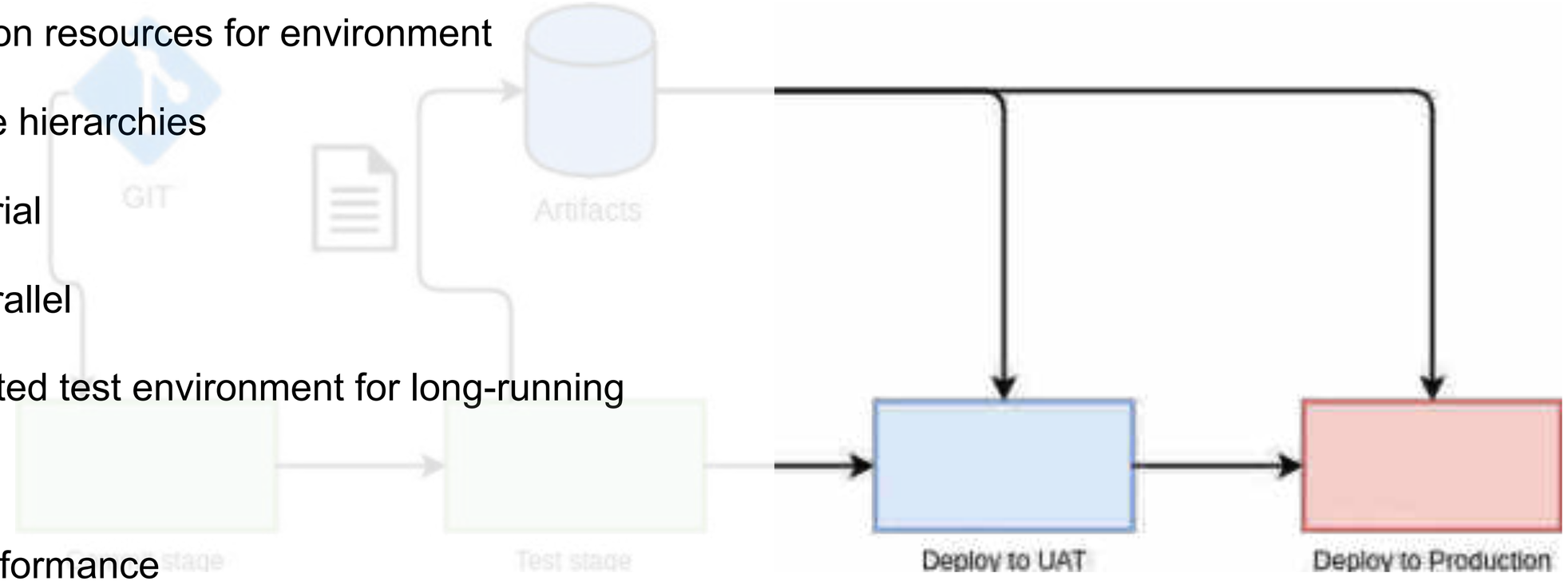
Multiple hierarchies

- Serial
- Parallel

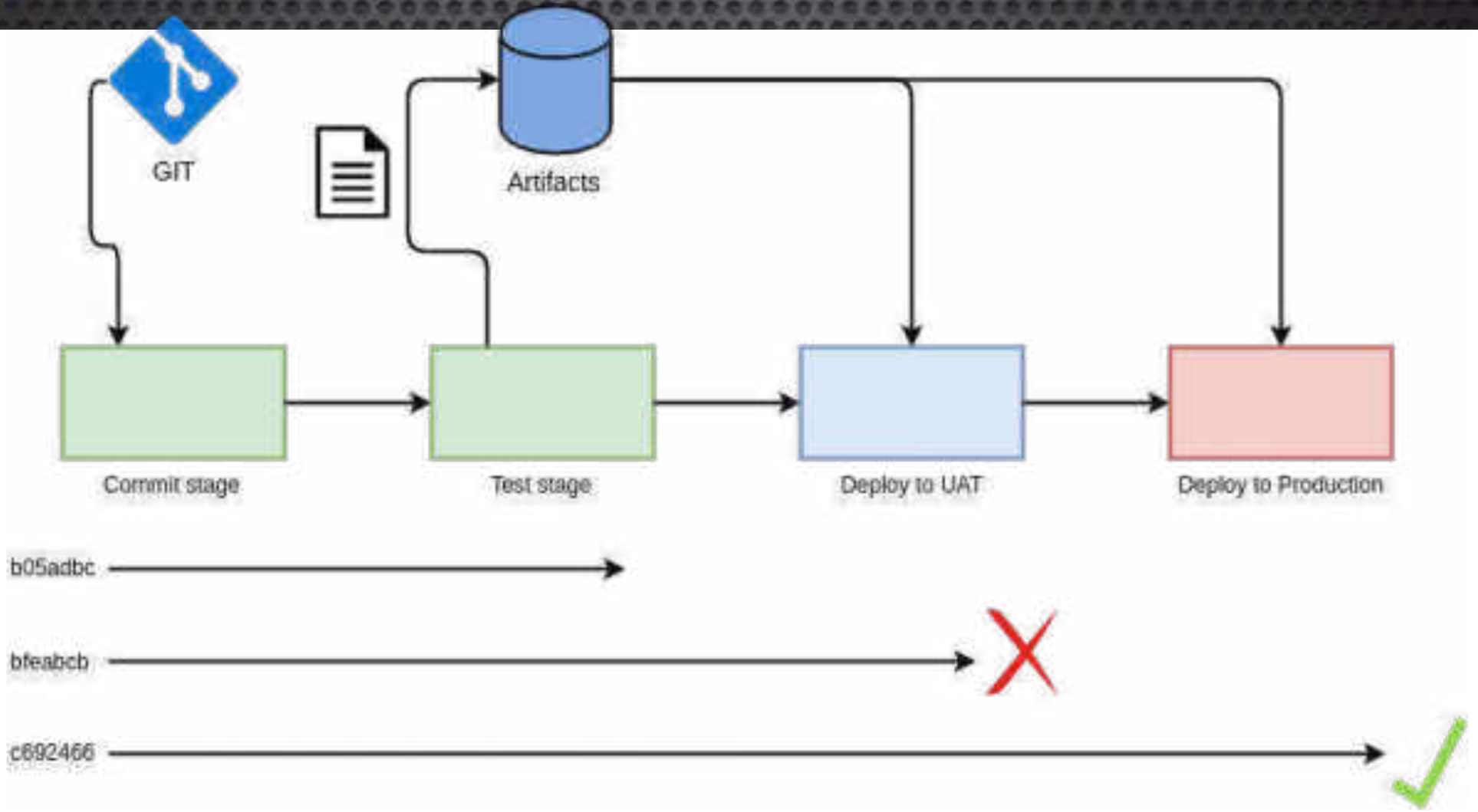
Dedicated test environment for long-running tests

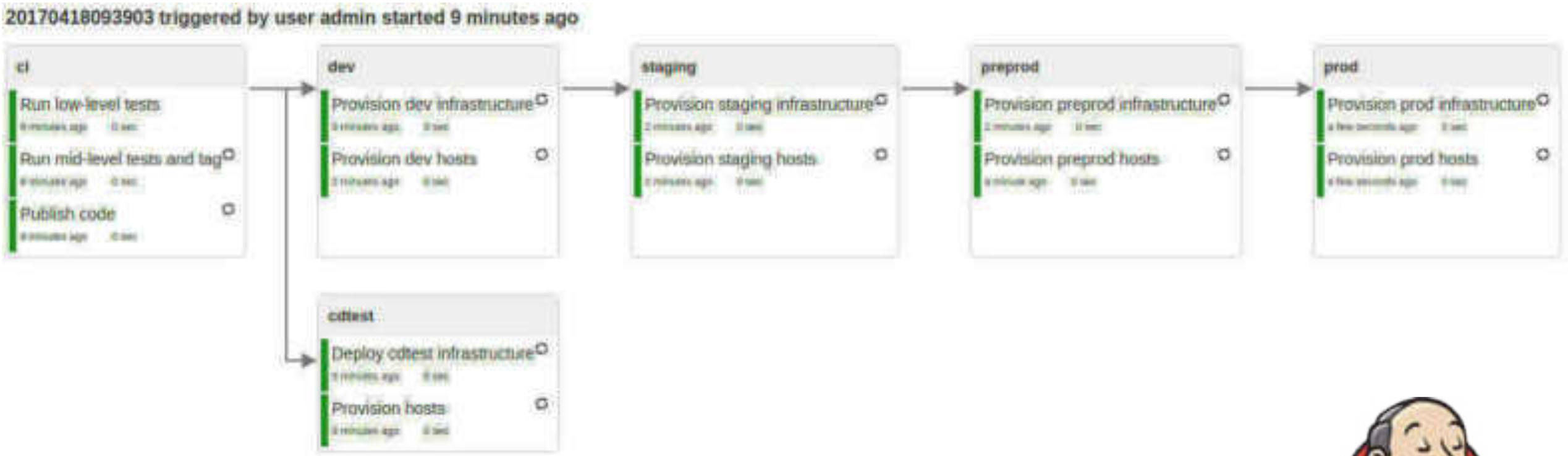
- performance
- compliance/security

Environment isolation levels (VPC, AWS account)



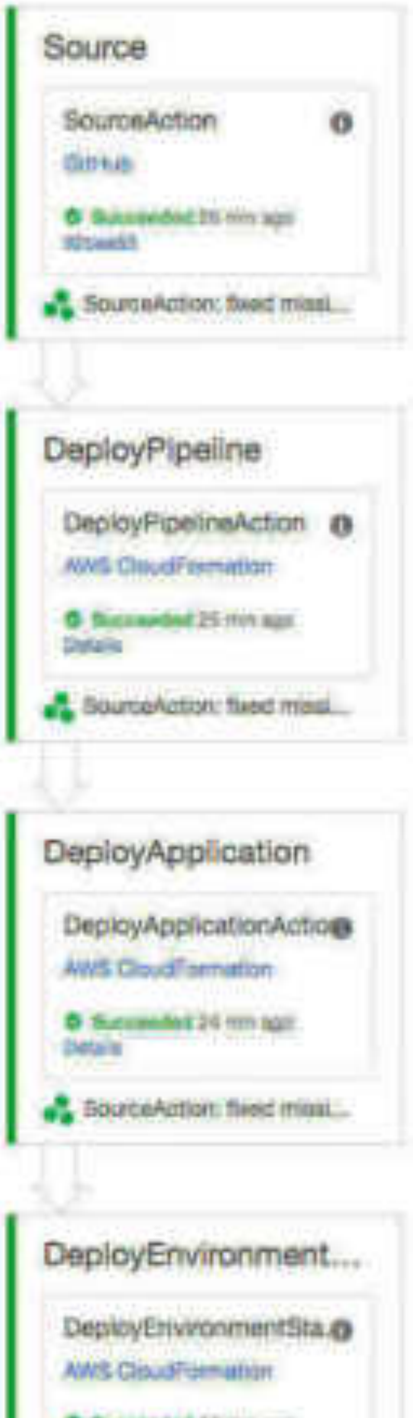
Pipeline


GIT
Artifacts
Commit stage
Test stage
Deploy to UAT
Deploy to Production
b05adbc
bfeabcb
c692466



Pipeline implementation on Jenkins

AWS CodePipeline + CloudFormation

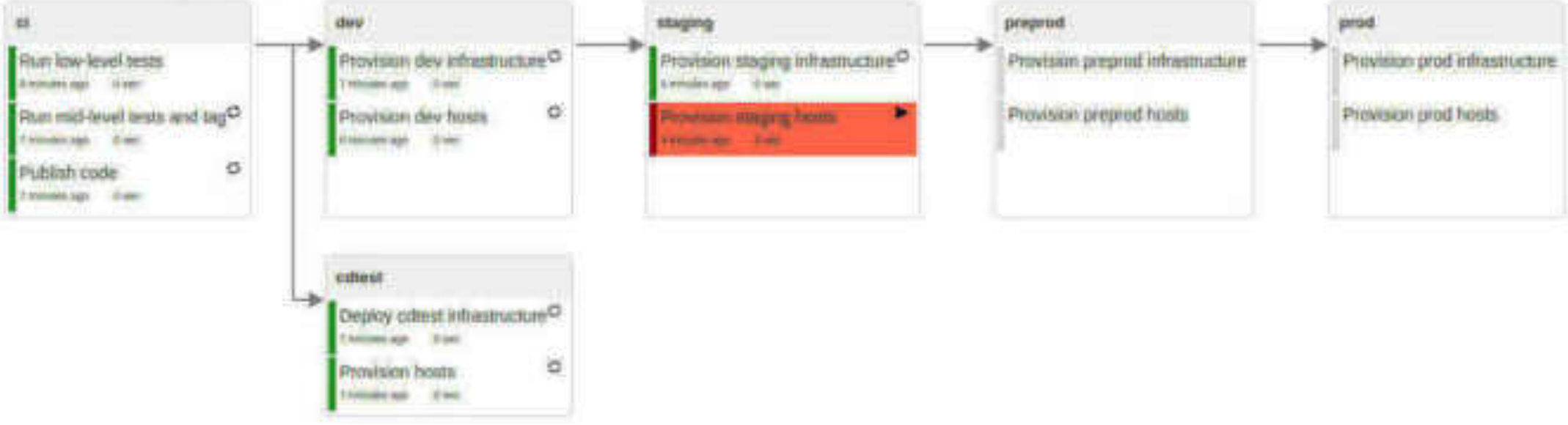


AWS Infrastructure Provisioning Pipeline

1

Aggregated view

20170418094147 triggered by user admin started 8 minutes ago



Implementation tips

Automation

Automation


I make changes outside my automation tool
My environment is incosistent
Fear!
I'm afraid that running my automation tool will break something

What if I'm not a programmer?

Become YAML, JSON "programmer"

Shell scripts are back!

Start learning to code - there's no way back!

Infrastructure as Code rules summary

1 - Focus on process, not tools

2 - Version all the things

3 - Prepare your code to create repeatable and consistent environments

4 - Make your infrastructure Antifragile by learning and continuously improving

5 - Don’t forget about proper testing

Thank You!

Contact me:

Email: librevo@librevo.pl

LinkedIn: https://www.linkedin.com/in/tomaszcholewa

Blog: http://cloudowski.com