DevOps

What is a DevOps Practice

To be Clear, It’s Not Dev vs Ops

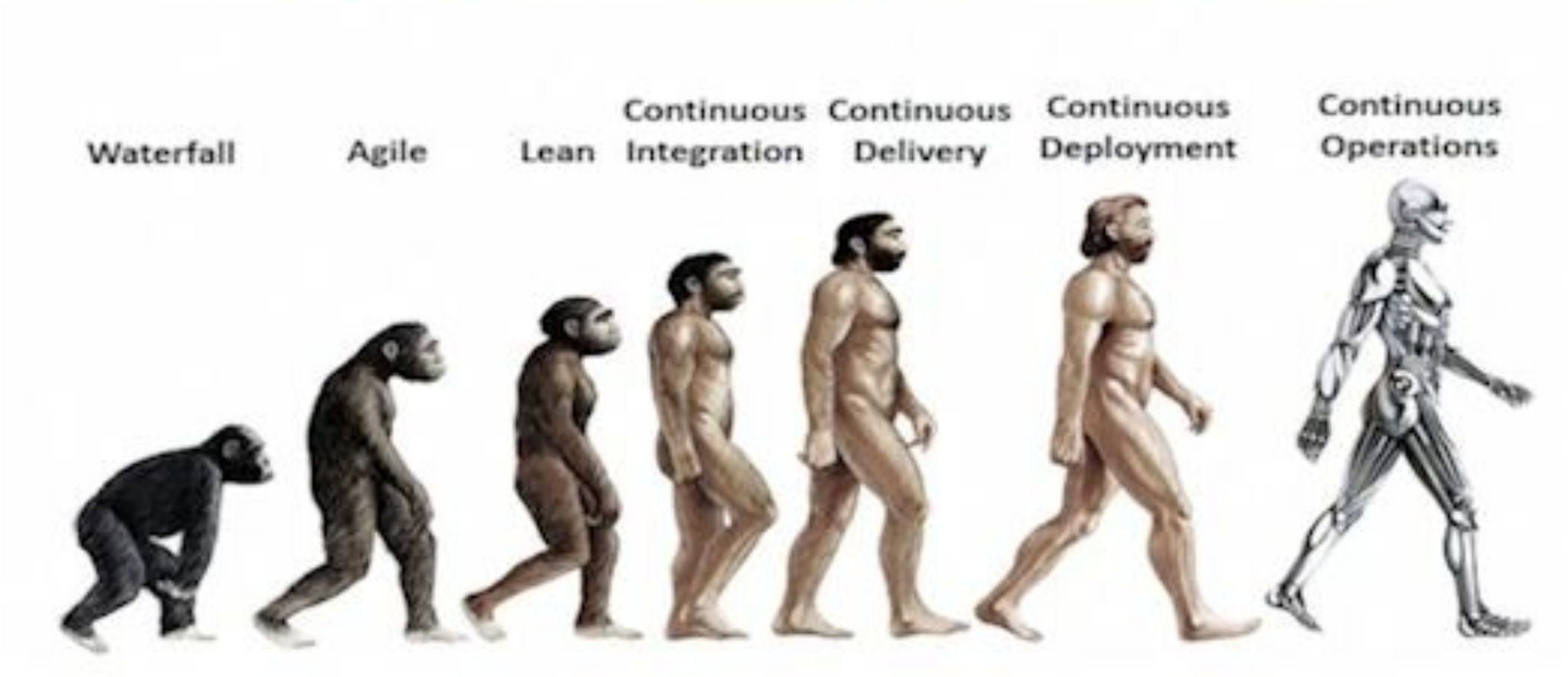
Definition

- DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

Where DevOps came From

- ITIL, ITSM, ESM, etc. underdeliver in IT from 1989 on
- Agile comes to the developer world in 2001
- Lean comes to the developer world in 2003 (more slowly)
- O'Reilly Radar "Operations: The New Secret Sauce" in 2006
- Agile Infrastructure discussions start in Europe circa 2007
- Patrick Debois and Andrew Schafer meet up at Agile 2008
- O'Reilly Velocity Conference starts 2008
- Velocity 2009, seminal John Allspaw "10+ Deploys Per Day: Dev and Ops Cooperation" presentation
- Patrick Debois and Kris Buytaert put together first DevOpsDays in Ghent in 2009. Many more follow
- Lean influences enter DevOps via startup culture
- Large companies start branding DevOps "solutions"

Evolution


Waterfall
Agile
Lean
Continuous Integration
Continuous Delivery
Continuous Deployment
Continuous Operations

DevOps timeline

DevOps Timeline

2007

Patrick Debois start's assessing IT Value Chain

2008

Agile System Administrators Group is launched on Google

Google

Agile2008 Conference

Andrew Clay Shafer and Patrick Debois meet at Agile Conference 2008

2009

Inaugural "DevOps Days" are held in Ghent, Belgium

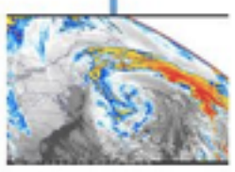
CHEF

Open Source toolsets rip up the legacy playbook.

O'REILLY Velocity Web Performance & Operations CONFERENCE

John Allspaw and Paul Hammond present "10 Deploys per day" at Velocity

2010



The "Perfect Storm" of adjacent methodologies occurs

2011

Industry leading software vendors increase market presence with "Enterprise" class DevOps tools

Industry leading software vendors increase market presence with "Enterprise" class DevOps tools

2012

IBM

hp

ca technologies

2013

Gene Kim releases "The Phoenix Project"

DevOps begins to provide positive impact to "Enterprise" IT

2014

devX is born and Xceed launches the "12 days of DevOps"

2015

What does the future of DevOps hold for your organisation?

Pillars

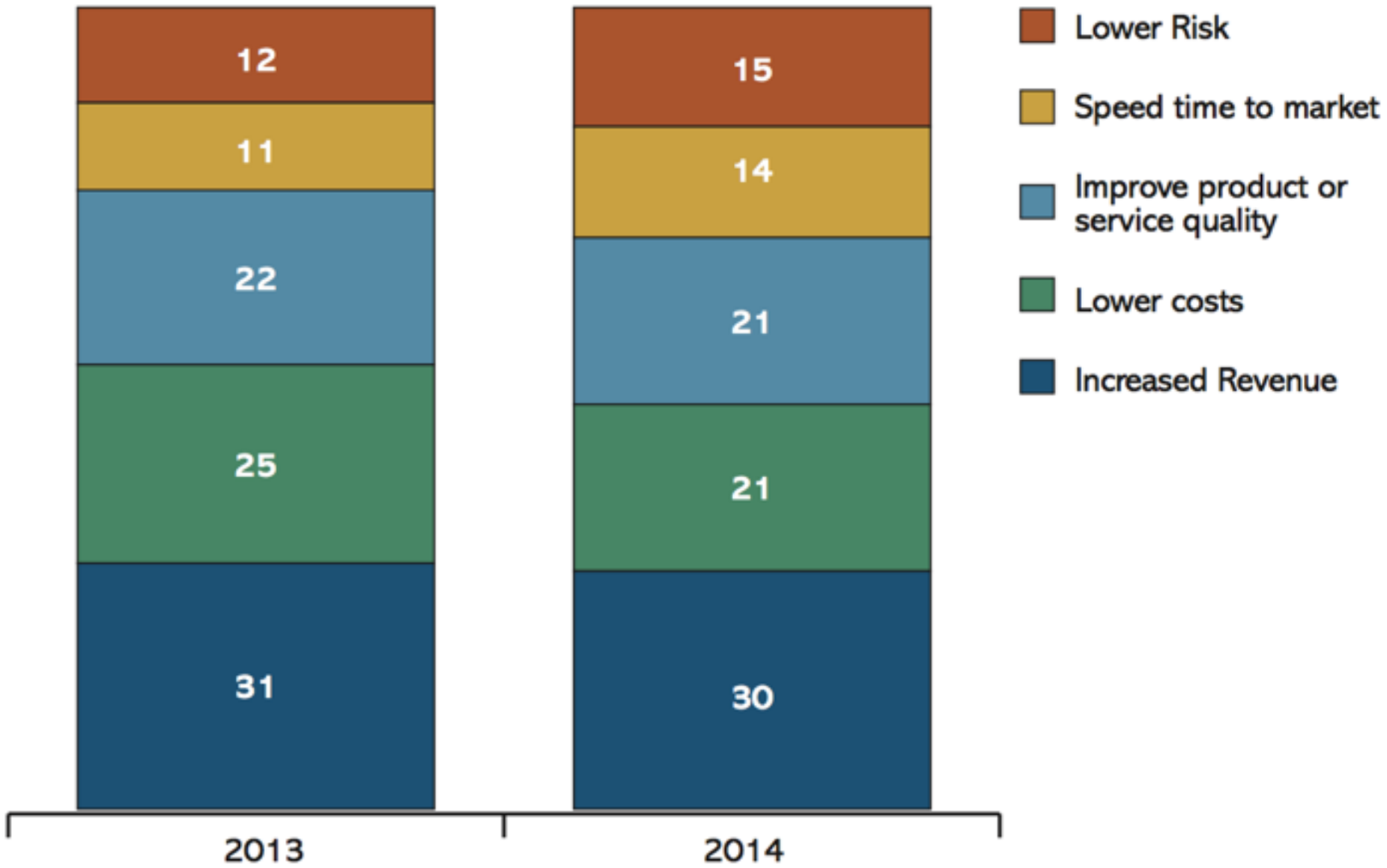
Operate
Support
Learn
Fix
Deploy
Process
Tools
Priorities
Skills
Development Operations
DEVOPS
Application Management
Configuration Management
Release Management
Break/fix
Fix
Design
Develop
Deploy
Test
Process
Tools
Priorities
Skills
Knowledge Transfer
Documentation 25%
On the Job Training 5%!
Class Room education 10%
Cross Training 30%!
Develop/Test

Lead Time
Shorten
Dev
Remove
Ops
Ah-ha!
Wall of Confusion
Feedback

Steer
Lean
Agile
Deploy
Operate

WSO2 Stratos
Project and Team Management
Software development workflow
WSO2 App Factory
Governance and Compliance
Development Dashboards
Develop Code
Issue Tracking
Source Control
Continuous Build
Continuous Integration
Test Automation
Continuous Deployment
puppet labs
JUnit
Se
sonar
EMMA
Bamboo
Maven
WSO2 App Factory
10/22/12

Minimize Risk, Maximize Agility

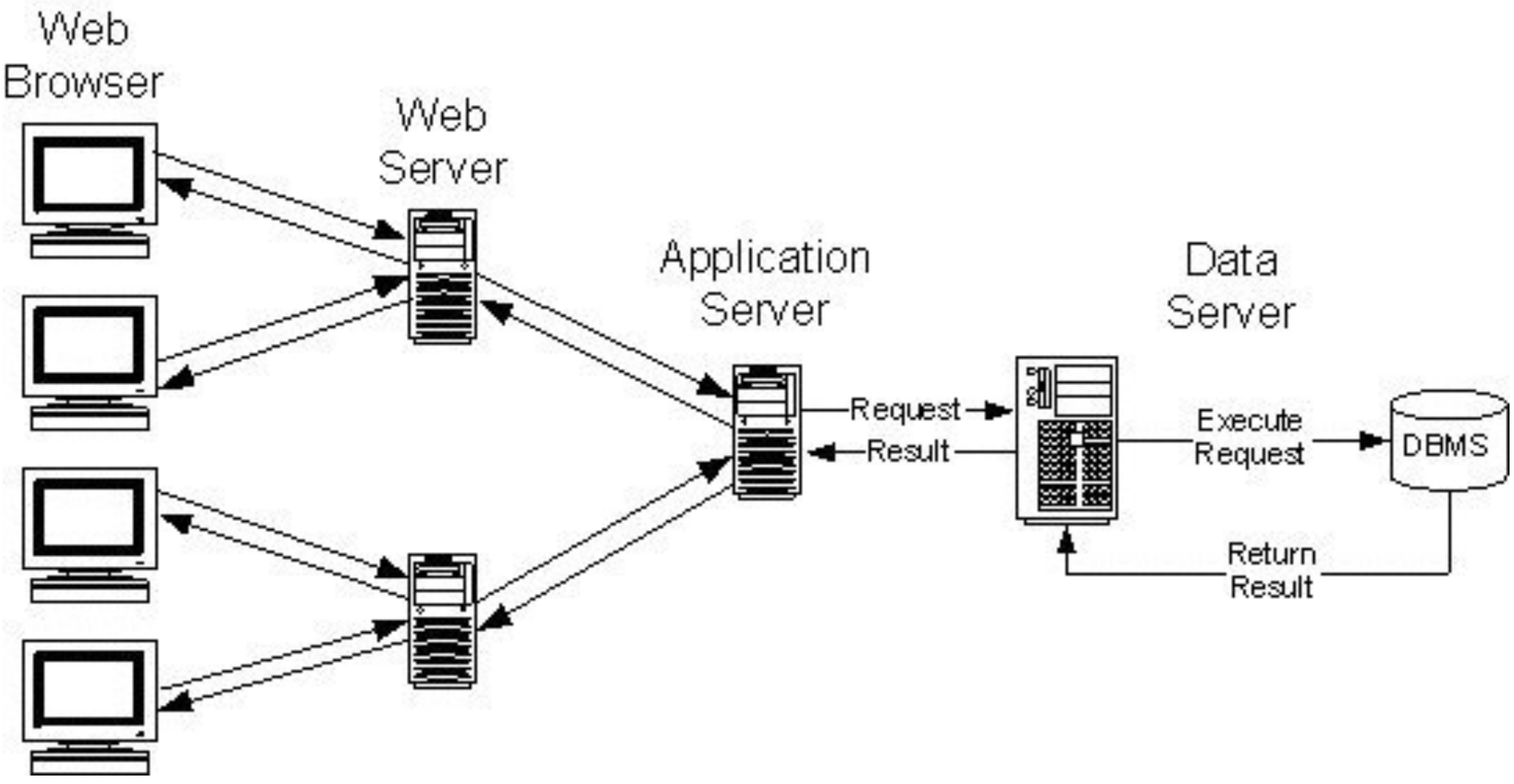


Why do it?

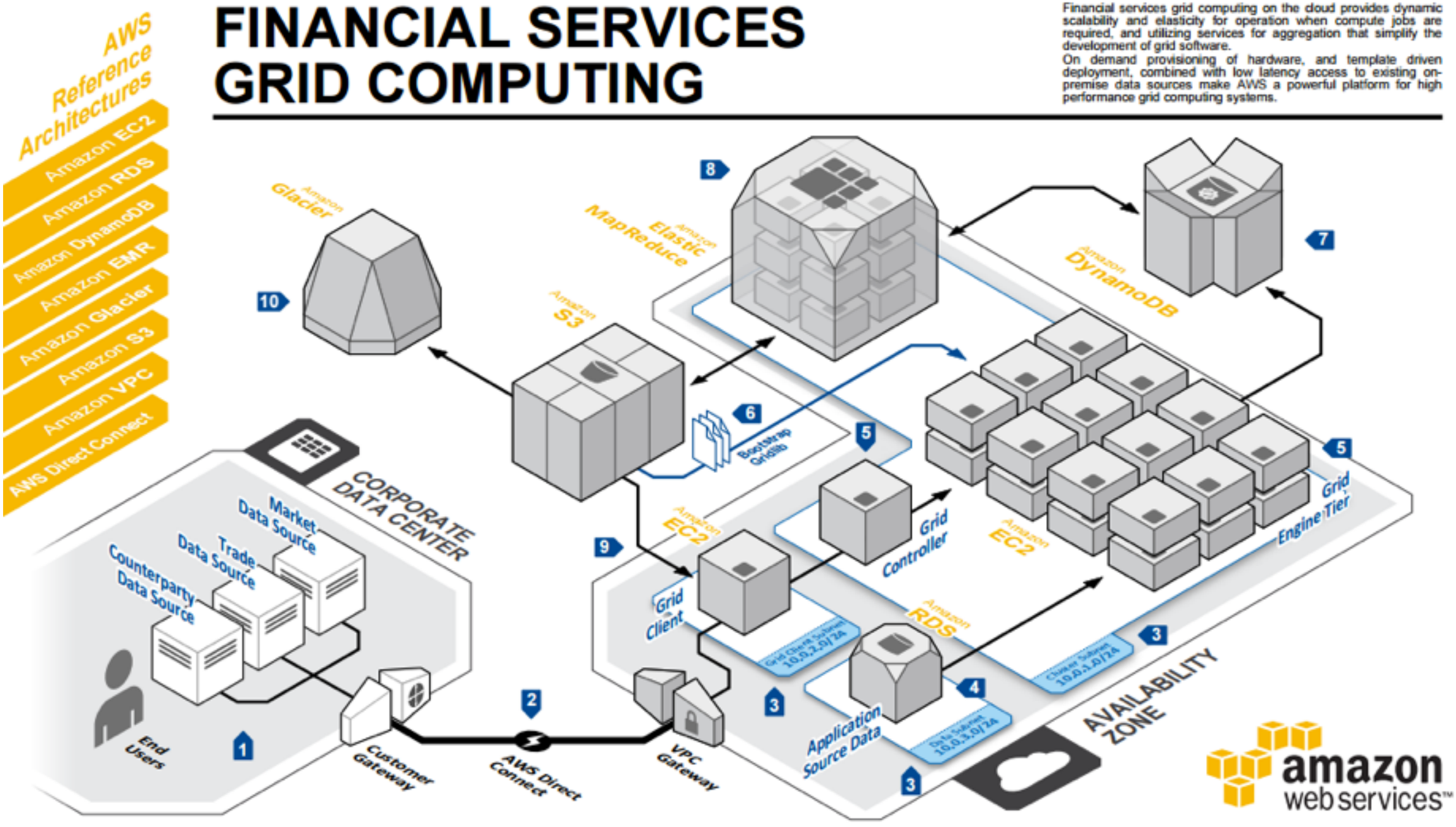
Why DevOps Now?

- Technological Advancements
  - Virtualization/Cloud
  - Commodization of Infrastructure
- Change Management Has Improved
  - Easier to change, faster to compensate
- Technology is no longer just a competitive advantage
  - It’s a necessity

From managing this…



To Managing this…



Looking at it another way….

- Defects released into production, causing outage
- Inability to diagnose production issues quickly
- Problems appear in some environments only
- Blame shifting/finger pointing
- Long delays while dev, QA, or another team waits on resource or response from other teams
- “Manual error” is a commonly cited root cause
- Releases slip/fail
- Quality of life issues in IT

The Essence of DevOps

> “Do painful things more frequently, so you can make it less painful…”

Adrian Cockcroft, Architect, NetFlix

The Business Value

The business Value

- “Does your company struggle with increased demands for shorter release cycles, with business leaders expecting weekly, daily or even hourly releases and updates?”

63% agree overall

Making Changes When It Matters Most

“By installing a rampant innovation culture, we performed 165 experiments in the peak three months of tax season.”

“Our business result? Conversion rate of the website is up 50 percent. Employee result? Everyone loves it, because now their ideas can make it to market.”

–Scott Cook, Intuit Founder

Who Is Doing DevOps?

- Google, Amazon, Netflix, Etsy, Spotify, Twitter, Facebook …
- Dynatrace, CSC, IBM, CA, SAP, HP, Microsoft, Red Hat, …
- GE Capital, Nationwide, BNP Paribas, BNY Mellon, World Bank, Paychex, Intuit …
- The Gap, Nordstrom, Macy's, Williams-Sonoma, Target …
- General Motors, Raytheon, LEGO, Bosche …
- UK Government, US Department of Homeland Security …
- Kansas State University…

Who else?

Continuous Delivery:
From dinosaur to spaceship in 2 years

Darren Hague / SAP Global IT
February 5, 2013

Public

Recap: Impact of Continuous Delivery in SAP Global IT

- **Before**: Production releases ~monthly
- **Now**: Production release ~twice a week
- **Before**: Pre-release QA cycle 1-2 weeks
- **Now**: QA cycle < 1 day
- **Before**: Error in Prod? Shitstorm & late night
- **Now**: Switch to Blue in <1 minute, fix next day
- **Before**: Project idea to go-live in 6-12 months
- **Now**: New project can be in Production in 1 week
- **Before**: Business stakeholders frustrated
- **Now**: Business stakeholders happy

Technology supports all this, but the team still has to deliver working code.

Source: Darren Hague (@dhague)

Value to business

- Engineering time is scarce
- Time is currency
- Move with the market

Knight Capital Group

DealB%k

ANDREW ROSS SORKIN
EDITOR-AT-LARGE

MERGERS & ACQUISITIONS | INVESTMENT BANKING | PRIVATE EQUITY | HEDGE F

LEGAL/REGULATORY | AUGUST 2, 2012, 9:07 AM | 357 Comments

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER

Brendan McDermid/Reuters

Results for IT

IT Results

Current IT operating models are not designed for today's high-velocity business environment

IT organizations typically spend more time testing, deploying and releasing software than designing and building it.

In a recent Forrester survey, just 17 percent of IT executives said they deliver fast enough for the pace of business.[1]

A high proportion of production incidents are a result of human errors in the manual release of software.

IT Development and IT Operations have different values and ways of working that are often not in alignment.

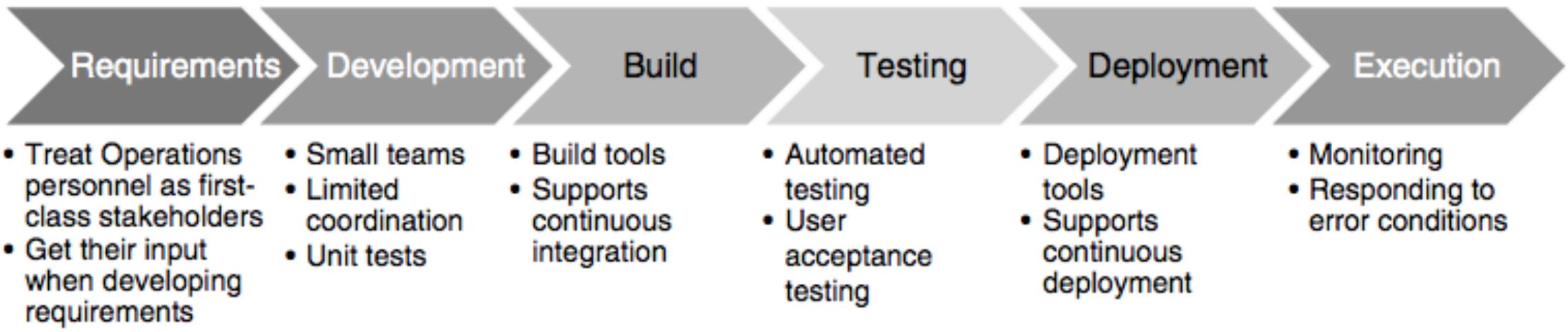
[1] "IT Speed: The Crisis and the Savior of the Enterprise," A Forrester Consulting study commissioned by Chef, December 2013

What IT Wants

- Actually get something done of value
  - Often it feels like like running in a circle
- Spend time on items of value and not repetitive tasks
- Speed - Tech is moving fast

A Real world Example

- Dev organization in a Product company
- Challenges
- The Devops Solution



DevOps enables IT delivery at the speed of business demands

CHALLENGES	HOW DEVOPS HELPS
Inefficient practices	Streamlined practices
Phased delivery	Continuous delivery
Manual processes	Automated quality processes
Siloed teams	Integrated teams

Gaps in Dev to Prod

- Service delivery is too slow and full of errors
- This loses you money. (Delay = loss)
- Therefore IT is frequently the bottleneck in the transition of “concept to cash.”

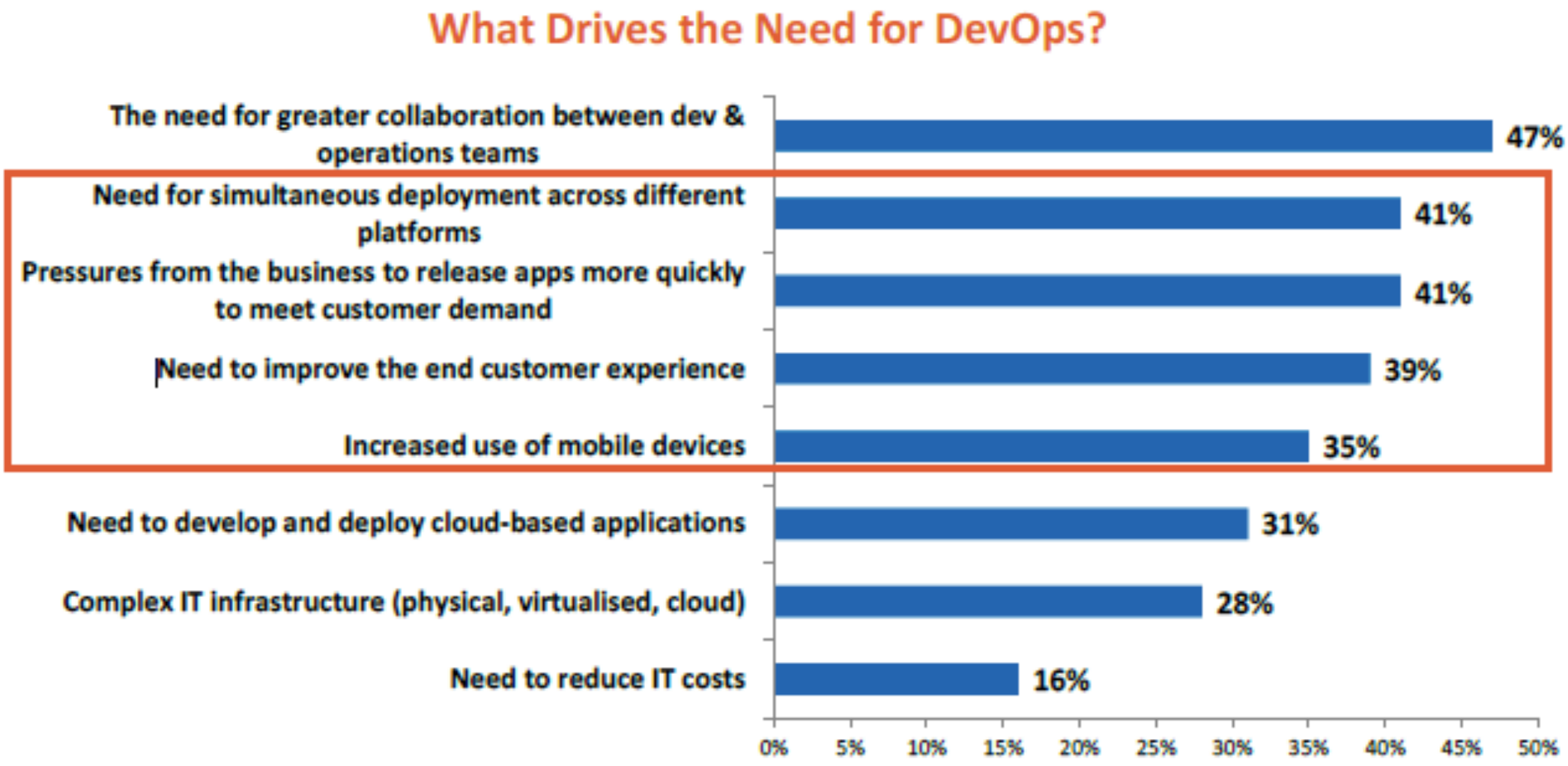
Enterprise DevOps

How to do this at scale

- Will require tools
  - Huge understatement
  - Cannot be manual
- But that is not all:
  - Shift in thinking about process
  - Choosing what people invest their time in. Culture shift.

Fighting Complexity

In General…



- We are tackling larger problems and solutions have become more complex

App Dev Complexities

- The infinite “stack” options
- Building modern systems has become incredibly complex
  - Dependency Management
  - Verifying it works
  - Number of developers involved
  - Deployment
- Ability to “try” different options quickly
  - sometimes you don’t know until you give it a try

Infrastructure

- The complexities of distributed systems
  - Deploying
  - Expanding
  - Managing
  - Making enhancements across the infrastructure

DevOps Principles

Why Does This Problem Exist?

- “Business-IT Alignment?”
- The business has demanded the wrong things out of IT
- IT has metastasized over time into a form to give the business what it’s said it wants

Principles

- Improve Continuously
- Automate everything
- Cohesive Teams
- Control the Source
- Deliver in Small increments
- Experiment often

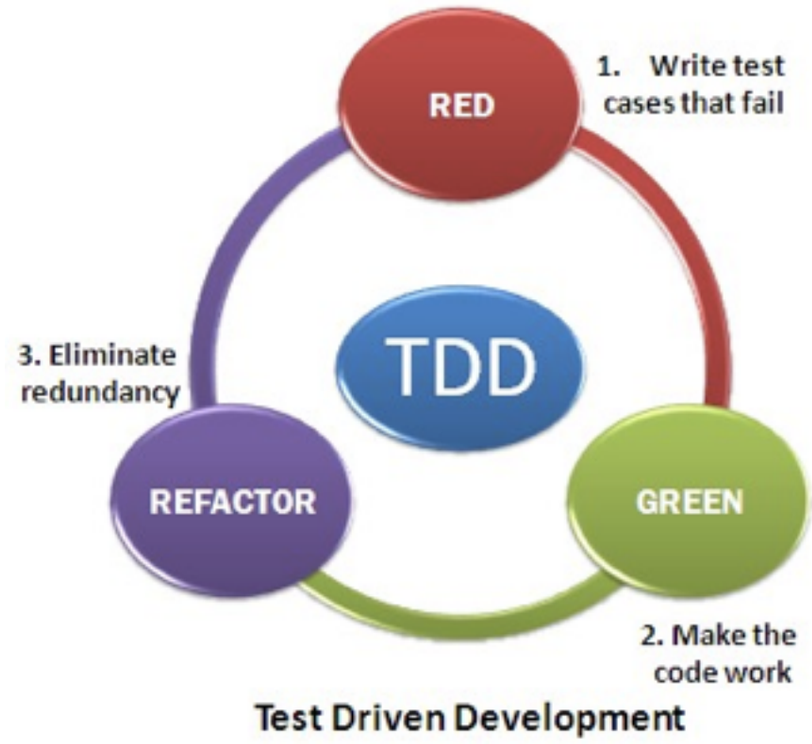
Devops and Others

Agile

- Tip to tail: Dev to Customer
  - Pain points
- The three pillars of the Agile stool
  - Automation
  - Testing
  - Skill
- DevOps fits like a glove

Dev/Test Cycle

- Zealots of automation
- Time is short - getting to market needs a well paved path
- Nothing manual
  - Especially QA

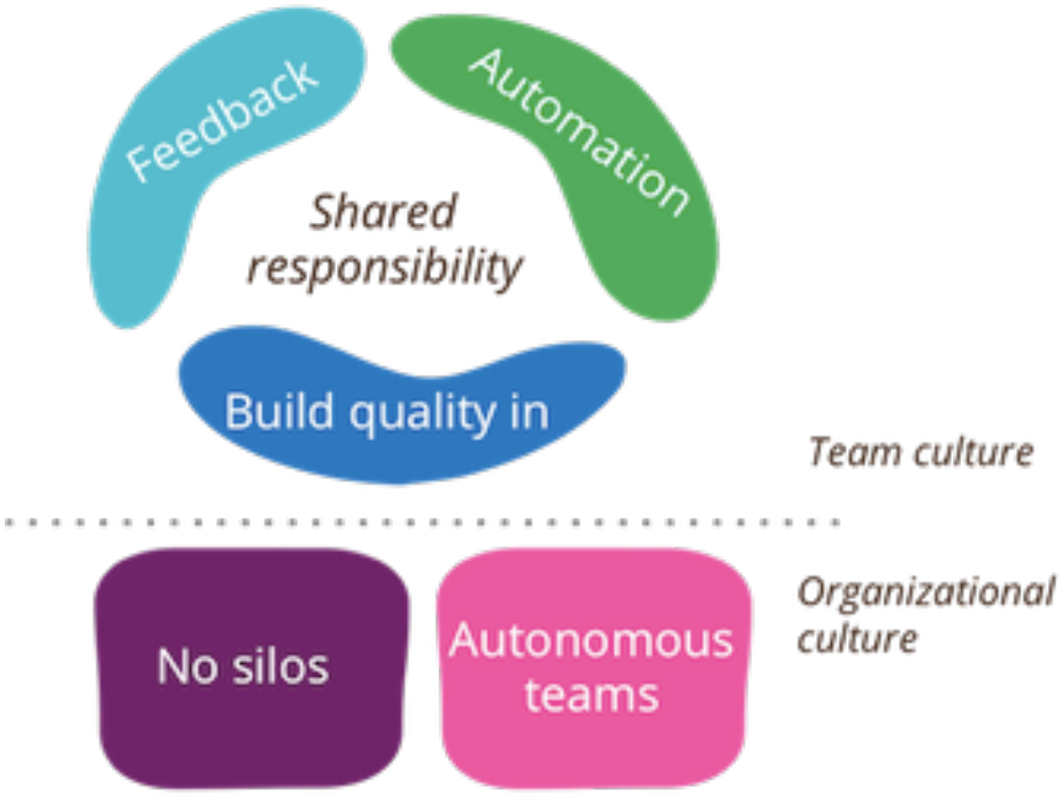


Test Driven Development

It takes a Culture

Culture

- Thinking in Systems
- Feedback Loops
- Continuous experimentation

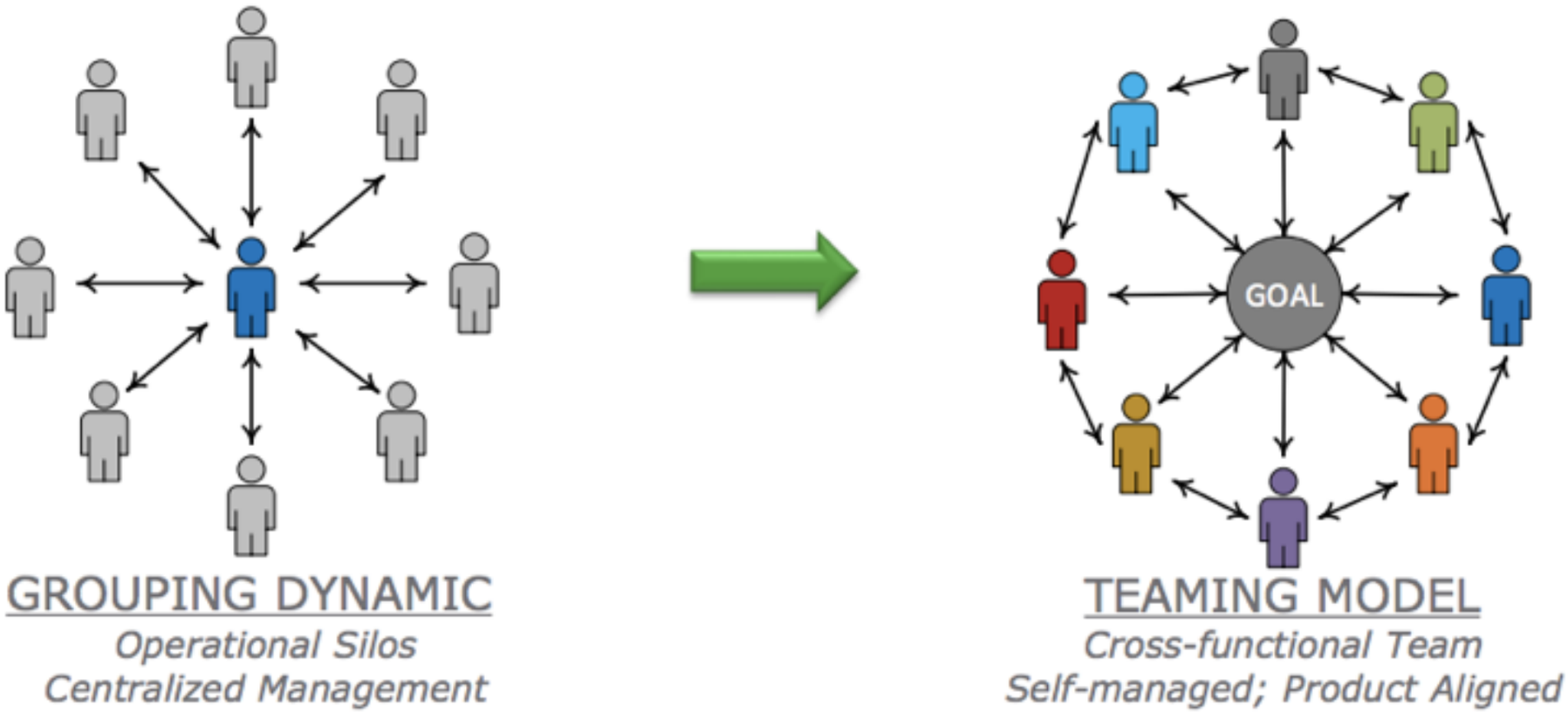


Foundations

- Version control
- Automated like a zealot
- Visibility
  - Everything in the open
  - Transparency
- Measure everything

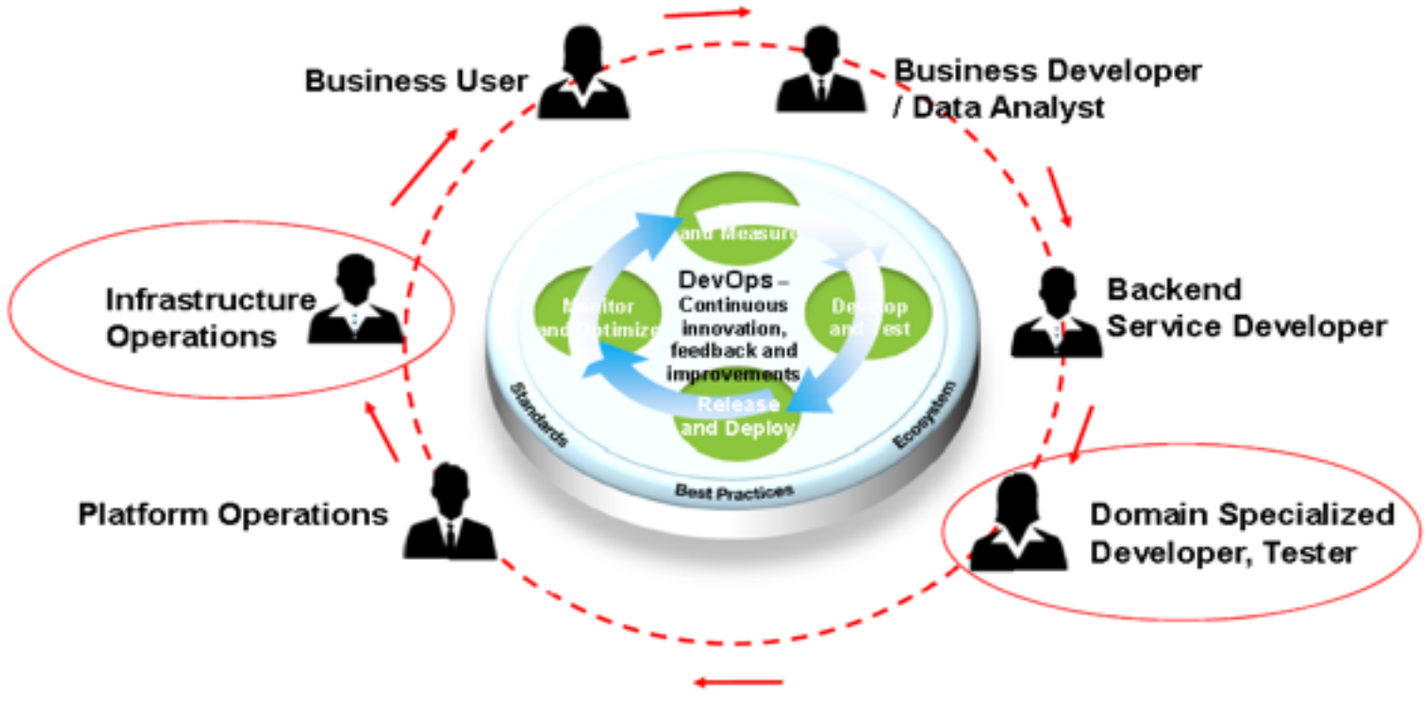
Organization Items

- From Silo to Open
  - Do not under-estimate
- Transparency required



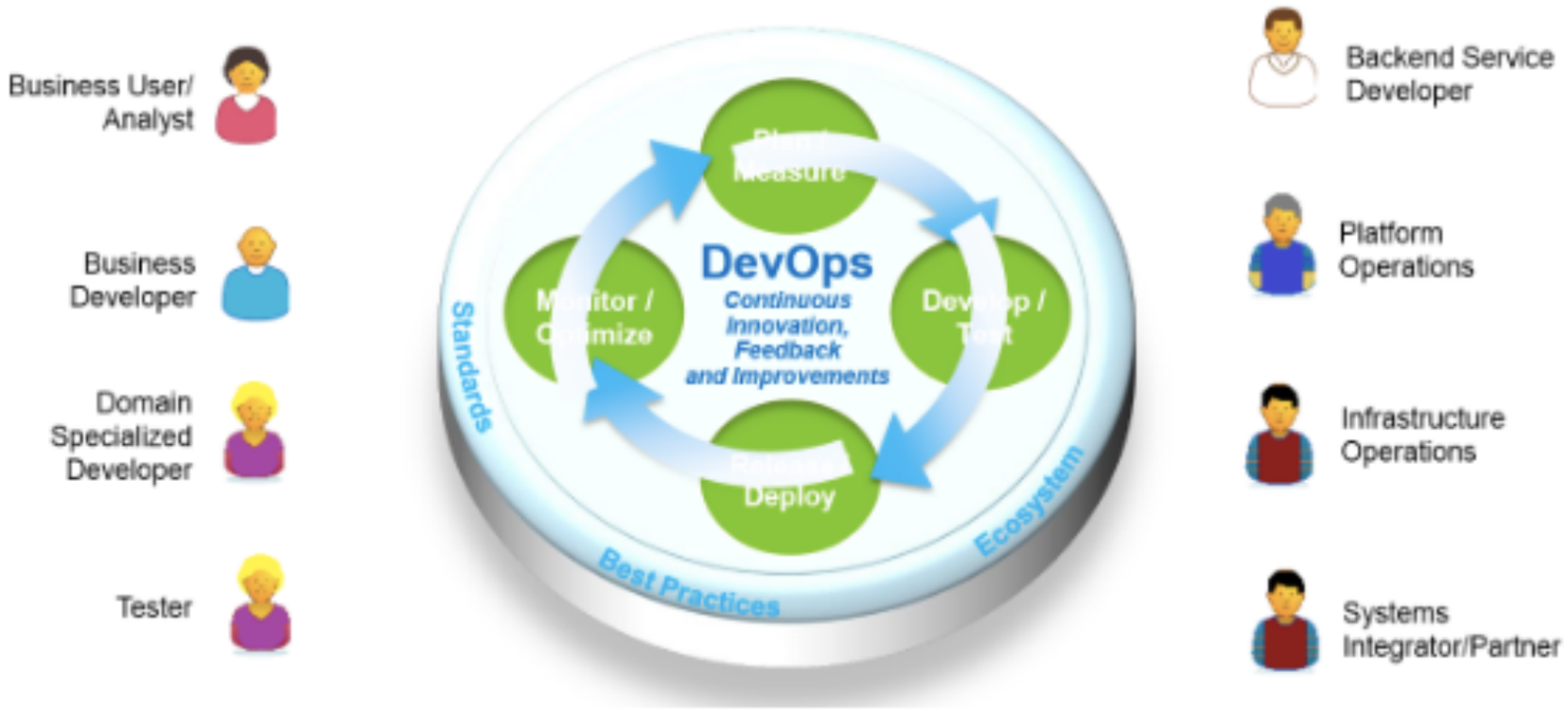
Stakeholders

- Who needs to buy in?
- Most DevOps movements start at the grassroots but will not succeed without management support



Roles

- Roles have to be re-adjusted
- Collapsed
- The Specialized Generalist



Practices

Create One Step Environment Creation Process

- Make environments available early in the Development process
- Common Dev, QA and Prod environment creation process

Breaking The Bottlenecks In The Flow

- Environment creation
- Code deployment
- Test setup and run (menti)
- Overly tight architecture
- Development
- Product management

“In November 2011, running even the most minimal test for CloudFoundry required deploying to 45 virtual machines, which took a half hour. This was way too long, and also prevented developers from testing on their own workstations.

By using containers, within months, we got it down to 18 virtual machines so that any developer can deploy the entire system to single VM in six minutes.”

— Elisabeth Hendrickson, Director of Quality Engineering, Pivotal Labs

Google Dev And Ops (2013)

- 15,000 engineers, working on 4,000+ projects
- All code is checked into one source tree (billions of files!)
- 5,500 code commits/day
- 75 million test cases are run daily

"Automated tests transform fear into boredom."
-- Eran Messeri, Google

Inject Failures Often

The Netflix Tech Blog

5 Lessons We've Learned Using AWS

We've sometimes referred to the Netflix software architecture in AWS as our Rambo Architecture. Each system has to be able to succeed, no matter what, even all on its own. We're designing each distributed system to expect and tolerate failure from other systems on which it depends.

One of the first systems our engineers built in AWS is called the Chaos Monkey. The Chaos Monkey's job is to randomly kill instances and services within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage.

You Don’t Choose Chaos Monkey…
Chaos Monkey Chooses You

The 2014 AWS Reboot

“When we got the news about the emergency EC2 reboots, our jaws dropped. When we got the list of how many Cassandra nodes would be affected, I felt ill.

“Then I remembered all the Chaos Monkey exercises we’ve gone through. My reaction was, ‘Bring it on!’”

Christos Kalantzis

Netflix Cloud DB Engineering

Source: http://techblog.netflix.com/2014/10/a-state-of-xen-chaos-monkey-cassandra.html

Add Ops Into Dev

- Enhance Service Design With Operational Knowledge
  - Reliability
  - Performance
  - Security
  - Test Them
- Build Feedback Paths Back from Production
- Foster a Culture of Responsibility
  - Whether your code passes test, gets deployed, and stays up for users is your responsibility – not someone else's
- Make Development Better With Ops
  - Production-like environments
  - Power tooling

- Time-to-Market
- Rapid Prototyping
- Experiment

Technical Innovation

- Enable the Polyglot
- Automate the Dev Chain
- Enable modern Architectures

Infrastructure Choice

- Containers
- Be Neutral
- Lightweight Hybrid Cloud with AWS, VMware, & OpenStack
- Consistent Design and Operations

Continuous Deployment

- Extension of continuous integration
- First: Automate deployment
- Application always known to be in a “deployable” state

Devops & IT Automation

Practices

One Step Environment Creation

- Need a common environment build process
  - For development, qa and production
- The environment will evolve as development proceeds
- The longer you wait to have a common environment build process, the harder it is to create one

Automating Feedback Loops

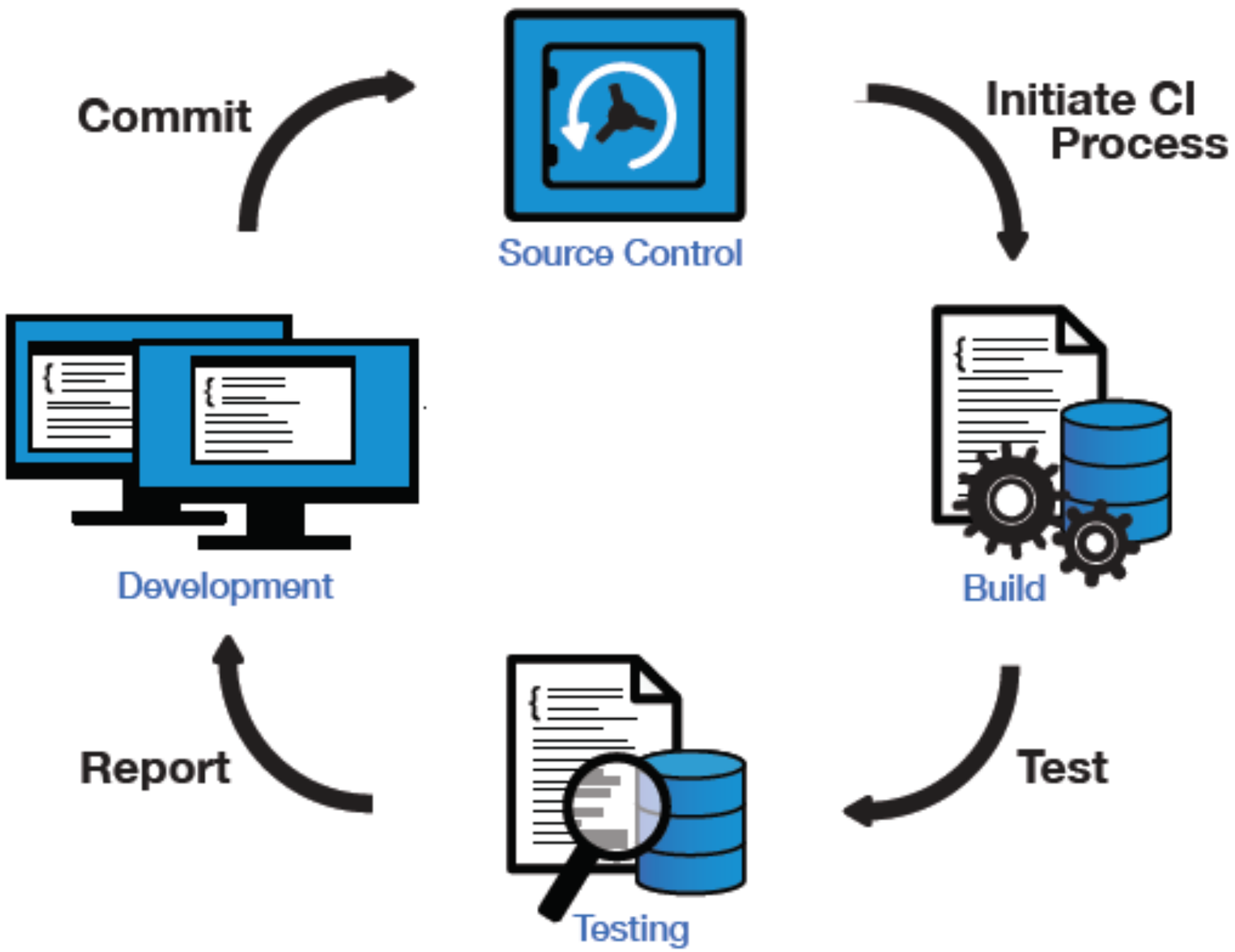
- Capture as much data as possible at the source
- The issue becomes the data

Break Things Early

- Consistency in code, environments and configuration
- ASSERTs to catch misconfigurations
- Static code analysis, and testing become part of the continuous integration and deployment

Lifecycles

Continuous Integration

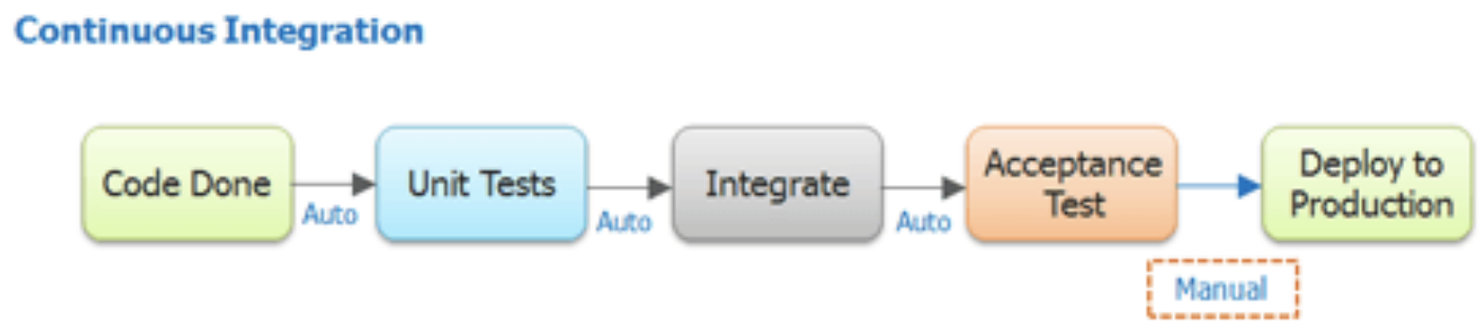


Principles of Continuous Integration

- Code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the main-line every day
- Every commit is built
- Build must be fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automated deployments

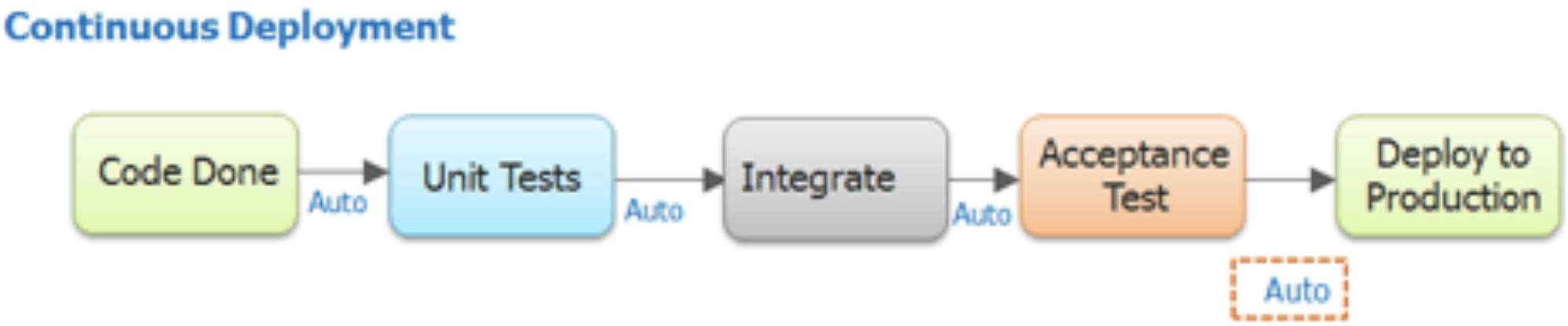
How CI Improves Efficiency

- Rapid Feedback
- Reduce technical debt
- Visibility
- Builds Automated
- Precursor to Continuous Delivery & Deployment



Keys to Continuous Delivery

- Minimize shock
- Avoid off-hour, high risk, expensive deployments
- Know your rollback plan
- Build in health checks



Feedback Loops

- Understanding and responding to the needs of all customers (internal and external)
- Shorten feedback loops
- Feedback = quality

Types of Tools

- Build tools
- Test tools
- General Automation
- Configuration Management

Adopting a Culture

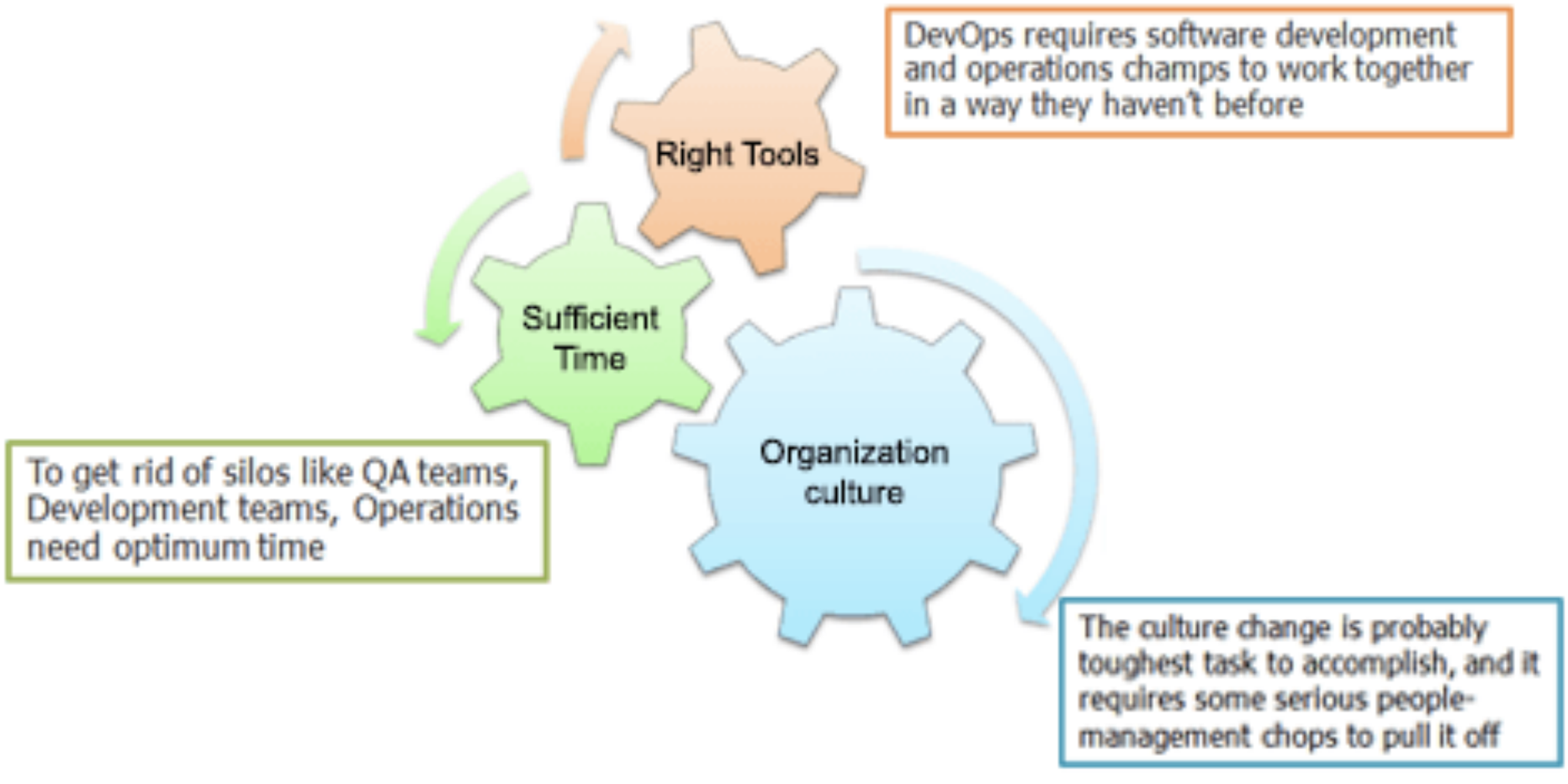
The 4 C’s


Reality Oriented
COLLABORATION
CONTROL
People Oriented
Company Oriented
CULTIVATION
COMPETENCE
Result Oriented

DevOps Culture Adoption

- Automate the Obvious
- Automate the painful
- Make it easy to do what is right
- Engineer with DevOps in mind
  - Tech select can impact the ability to make change easy

Working in a new way



Management Checklist

- Experiment – choose a test case as a pilot
- Then document and spread best practices
- Empower your teams, but guide their values
- Metrics are your friend – demand measurable outcomes
- Don’t accept excuses when the old baseline isn’t good enough
- Fail fast, continually improve
- Build on small successes to gain broad support for more substantive change.
- Align roles and responsibilities across groups –enable collaboration even if it seems “inefficient”

Challenges/Risks

- But we have not done it that way
  - The story of the manual QA team
- Grass roots
  - With a solid sponsor
- Cost of Adoption
  - Time
  - Monetary

Managing Change

"Leading Change" by John Kotter

Measurements

- How are we doing?
- Objective measurements
  - Dashboards (transparency)
  - Build time
  - Defects
- Feature time to market
- Measure everything

Related Tech

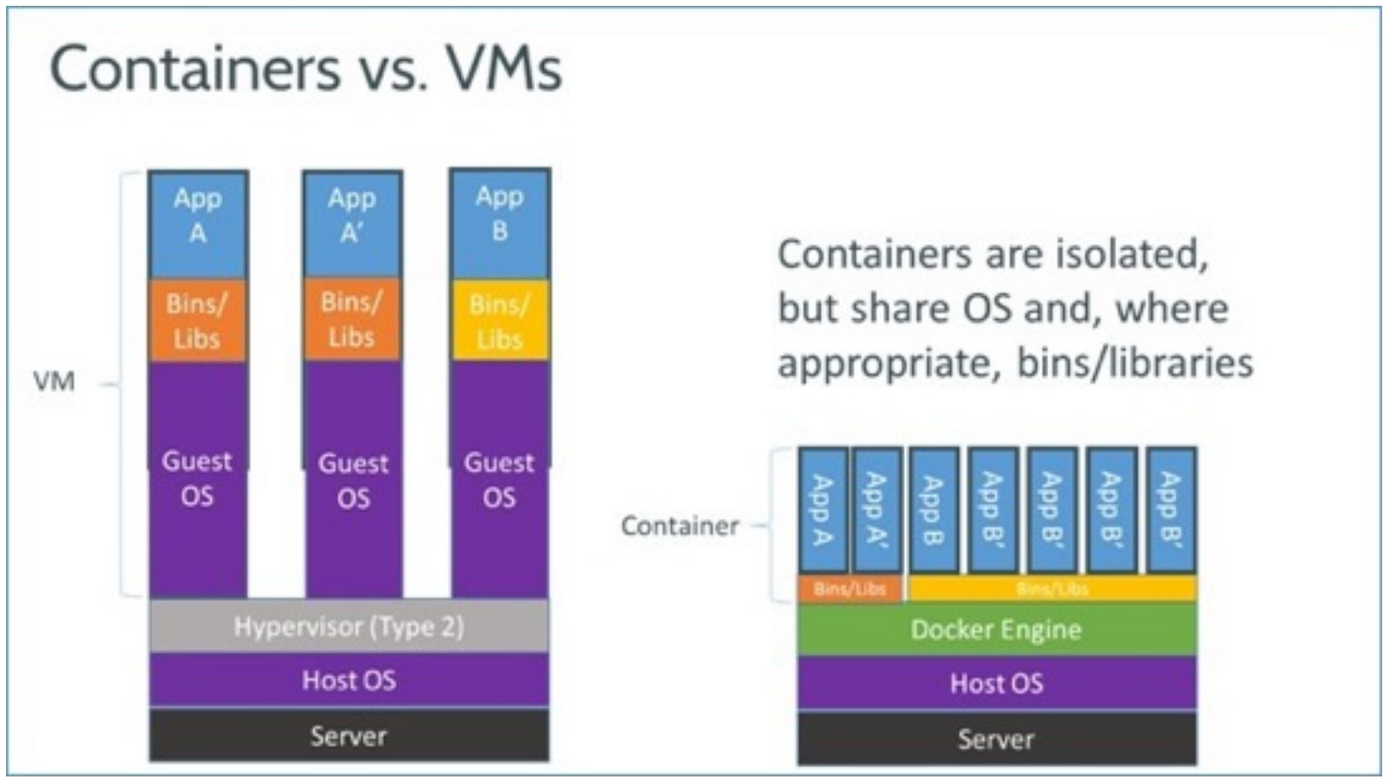
Virtualization

- Been a round for a while
- Think VMWare ESX
- Virtual Machines

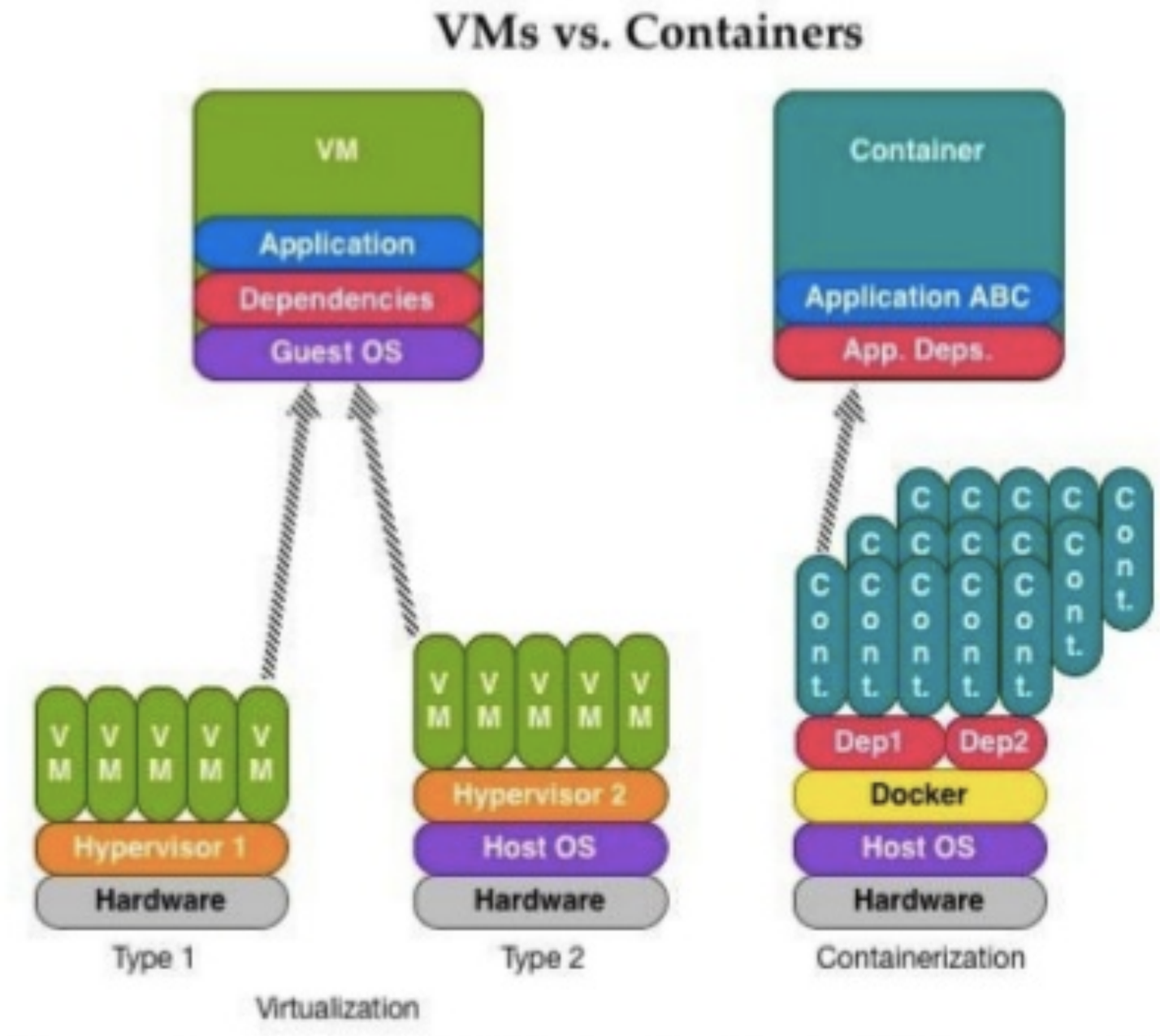
Containers (Docker)

- Open-source engine that automates the deployment of any application as a lightweight, portable, self-sufficient container that will run virtually anywhere.

- Based on LXC (Linux Container), and AUFS (Union FS), easy to use.
- Similar to VM as end-user with different features



Docker – what is it?



Current Container Issues

- Networking
- Service Orchestration
- Security

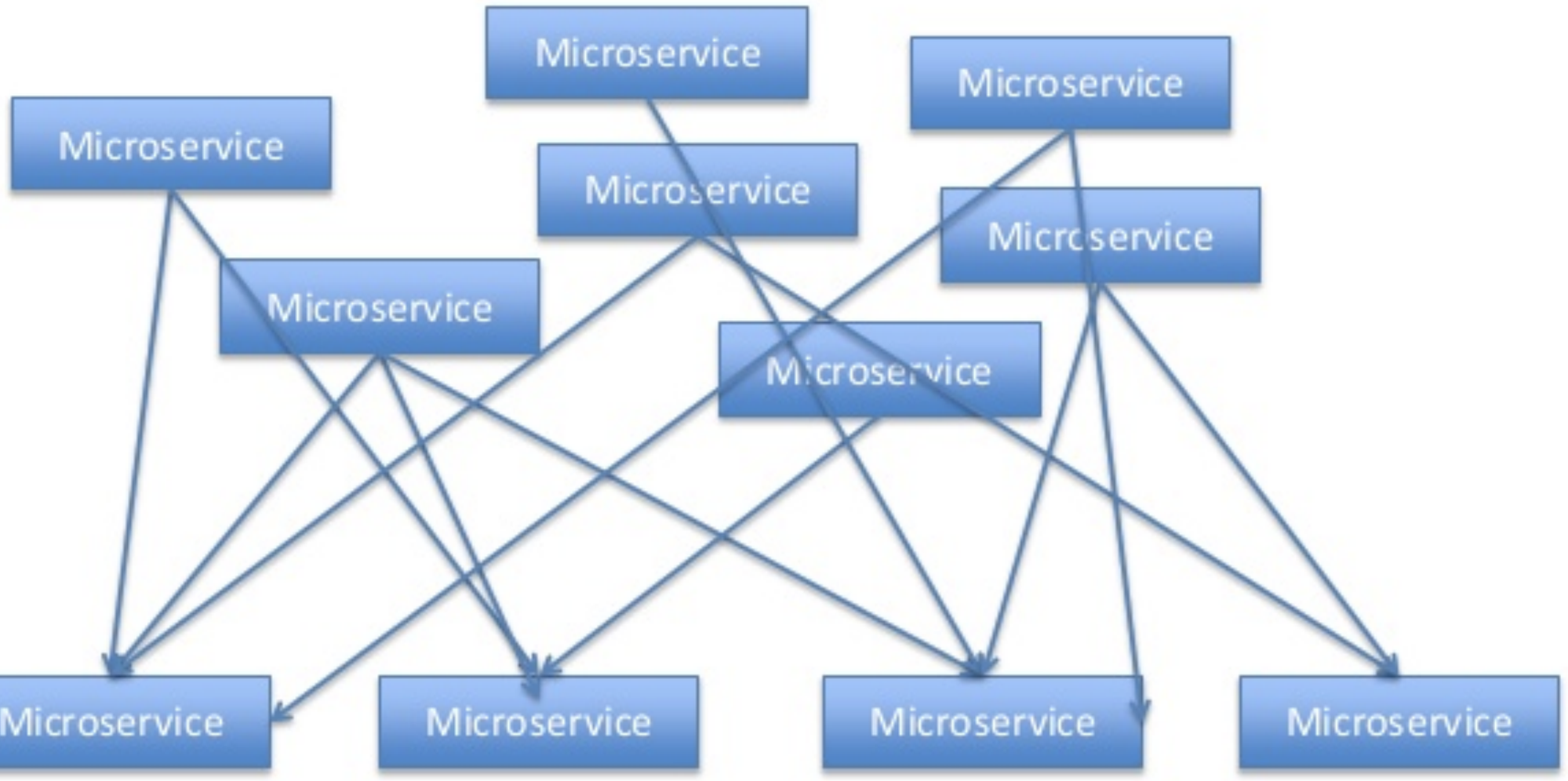
Docker Alternatives

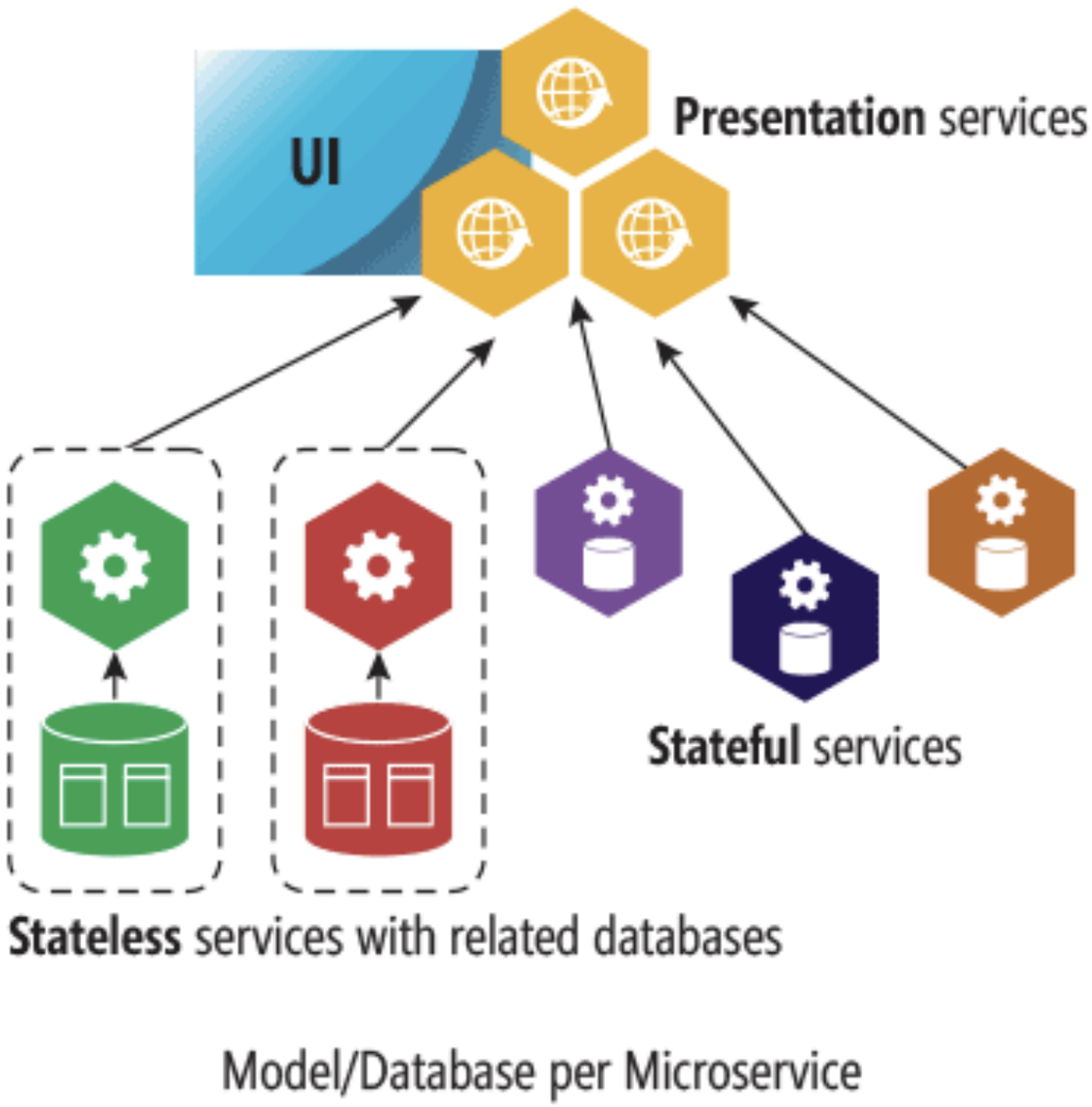
- Rocket
- Microsoft Drawbridge
- LXD (Canonical)

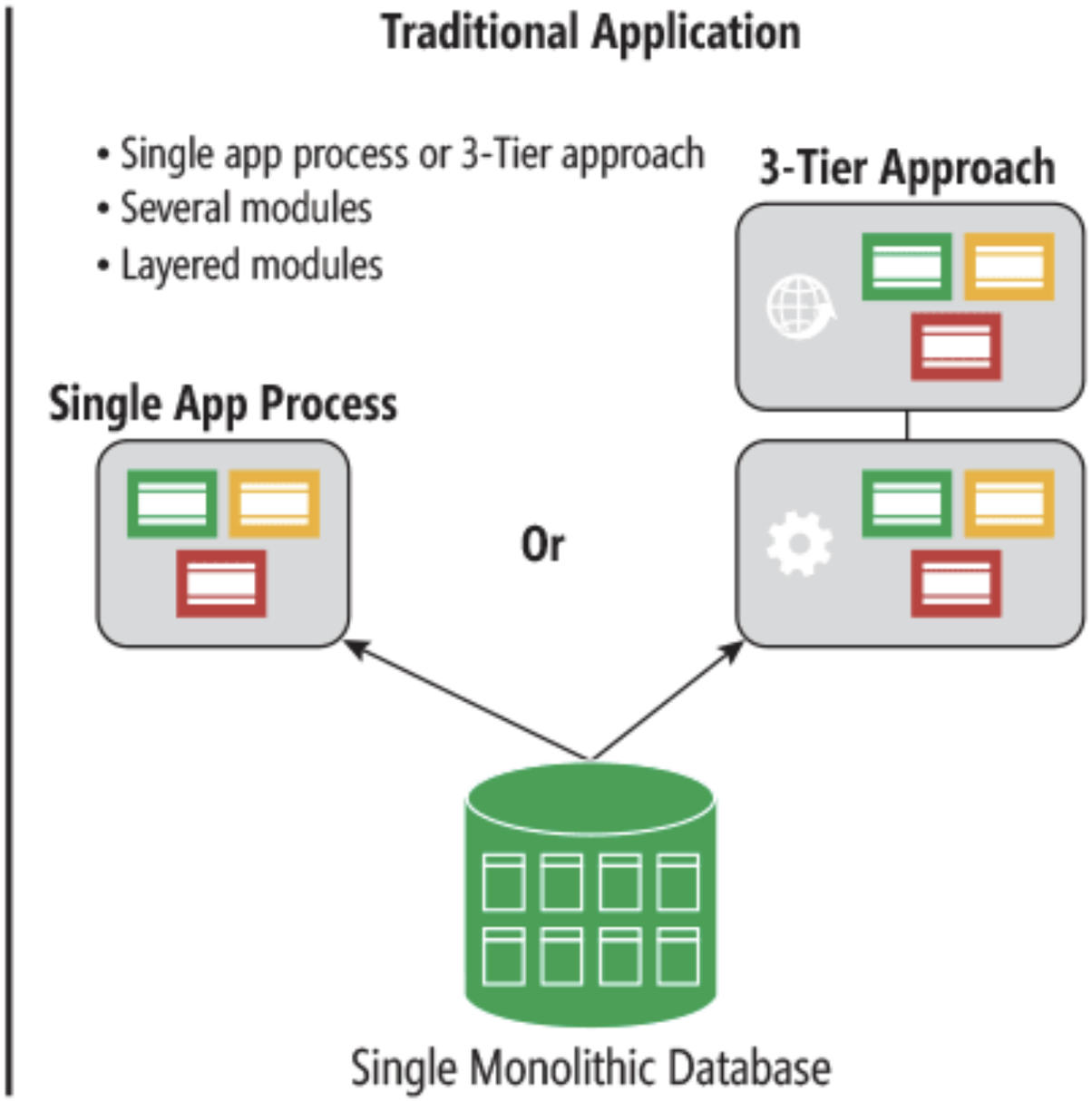
Microservices

- Architectural pattern based on a stateless service model
  - Granular services targeted at providing a discrete piece of application functionality
- Couples with the container model promoted by Docker
- Services are elastic and provisioned on demand in a container as needed

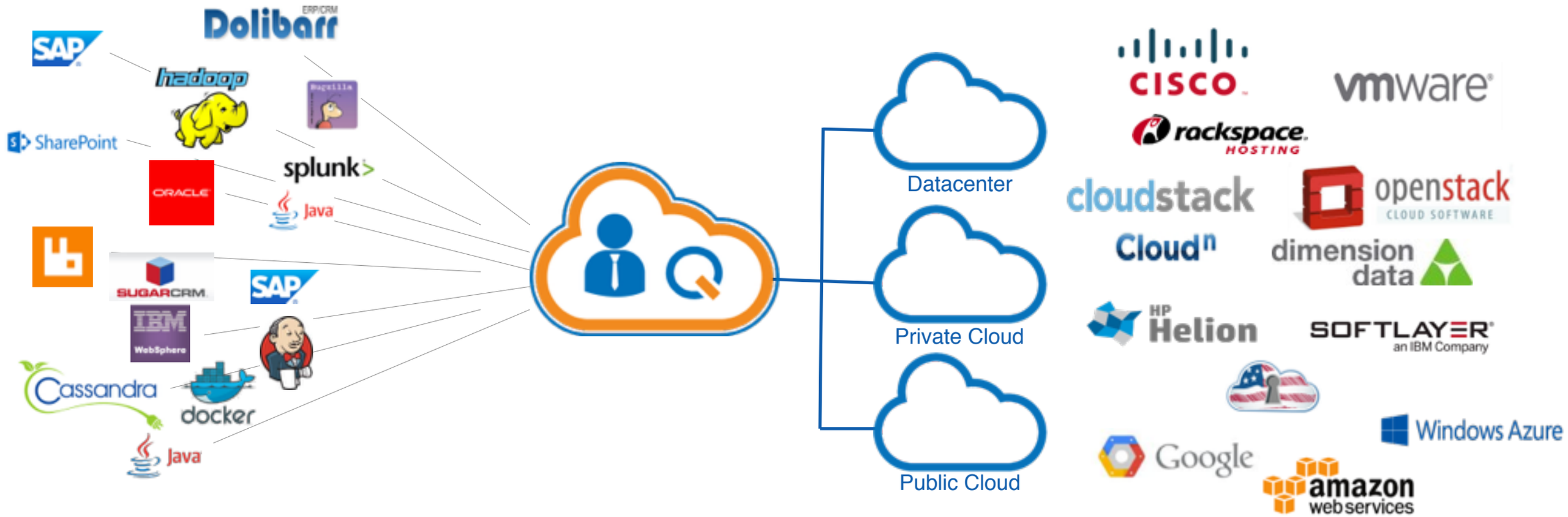
Distributed Service Layer
Microservices Architecture




Microservices Approach
UI
Presentation services
Stateful services
Stateless services with related databases
Model/Database per Microservice


Traditional Application
• Single app process or 3-Tier approach
• Several modules
• Layered modules
3-Tier Approach
Single App Process
Or
Single Monolithic Database

Any Application. Any Cloud. One Platform



Common Use Cases

Application Migration & Management

- New and existing applications
- Start with single application, single cloud
- Workload migration. Datacenter consolidation

DevOps and Continuous Delivery

- Self-service on demand environments
- Tool-chain automated deployment
- Hybrid cloud pipeline

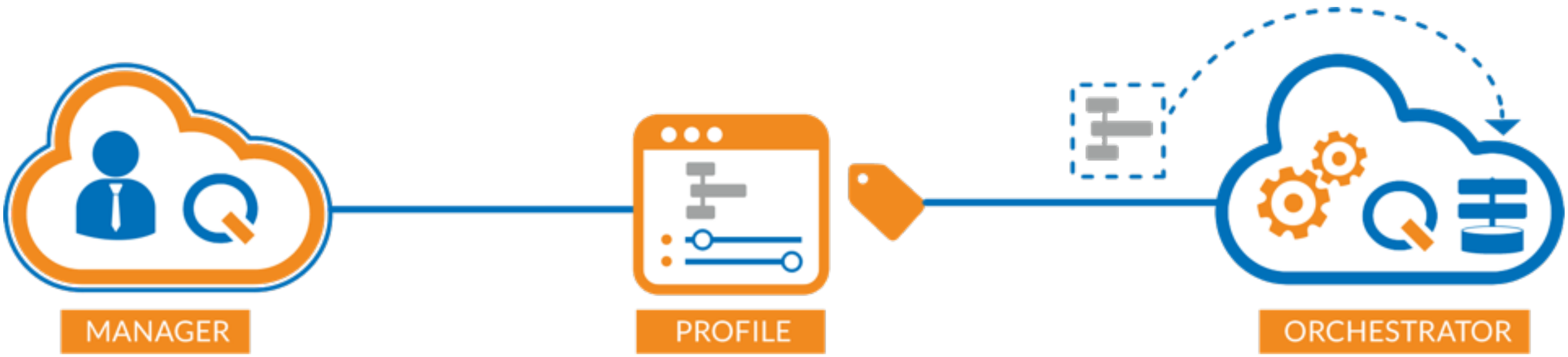
Dynamic Capacity Augmentation

- High performance, batch and cluster
- Horizontal scaling
- Cross-environment bursting

IT as a Service With Governance

- On-demand – self-service marketplace
- Single pane of glass - financial controls, usage metering, multi-tenant governance

CloudCenter



CloudCenter

- ✓ Single Platform Solution
- ✓ Unique application-defined technology
- ✓ Abstracts application from underlying cloud environment
- ✓ Ensures infrastructure adapts to meet the needs of each application

No need to:

- ✓ Write cloud specific scripting
- ✓ Write orchestration workflows
- ✓ Modify application code

SaaS
Software as a Service

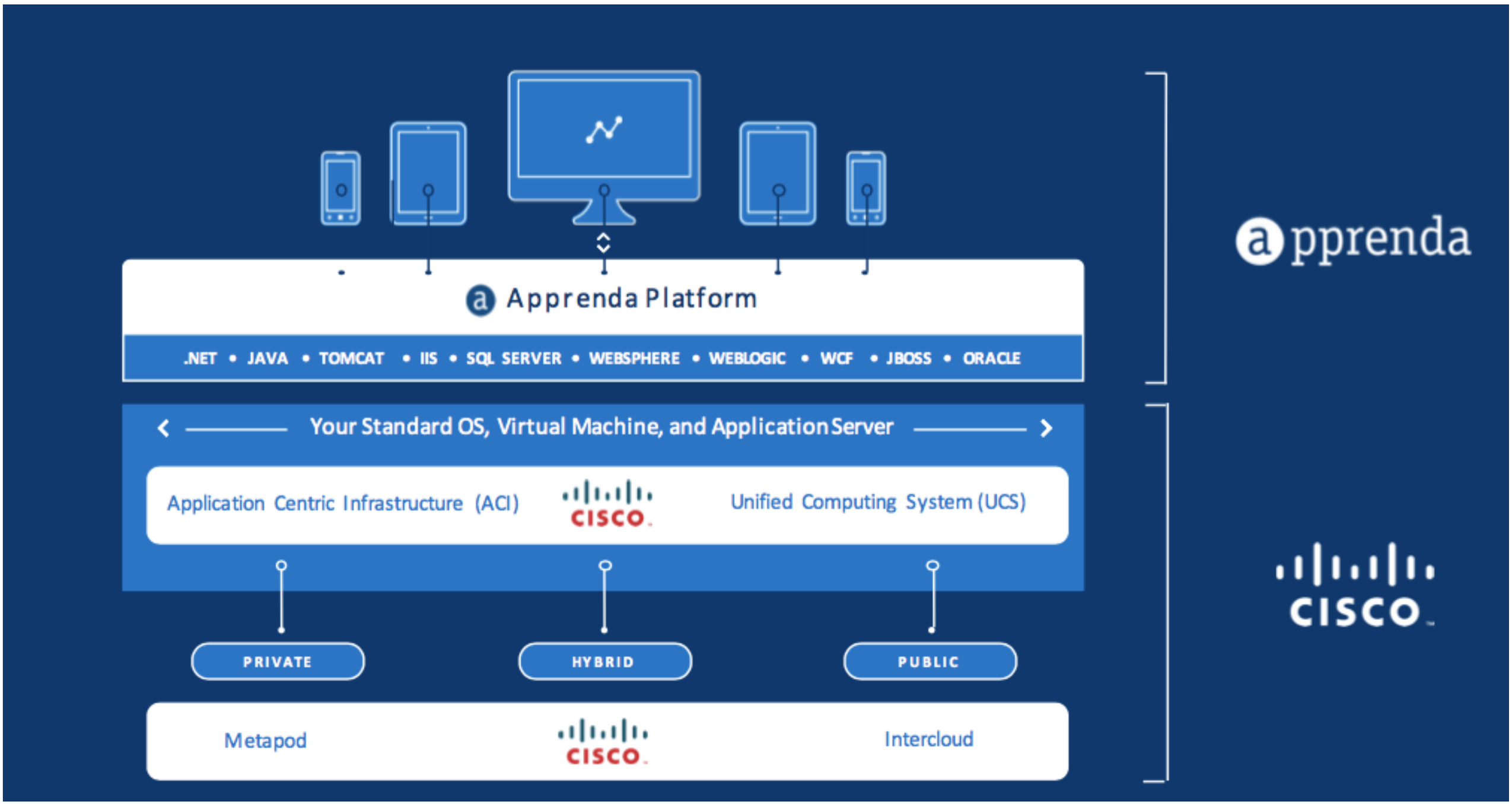
PaaS
Platform as a Service

IaaS
Infrastructure as a Service

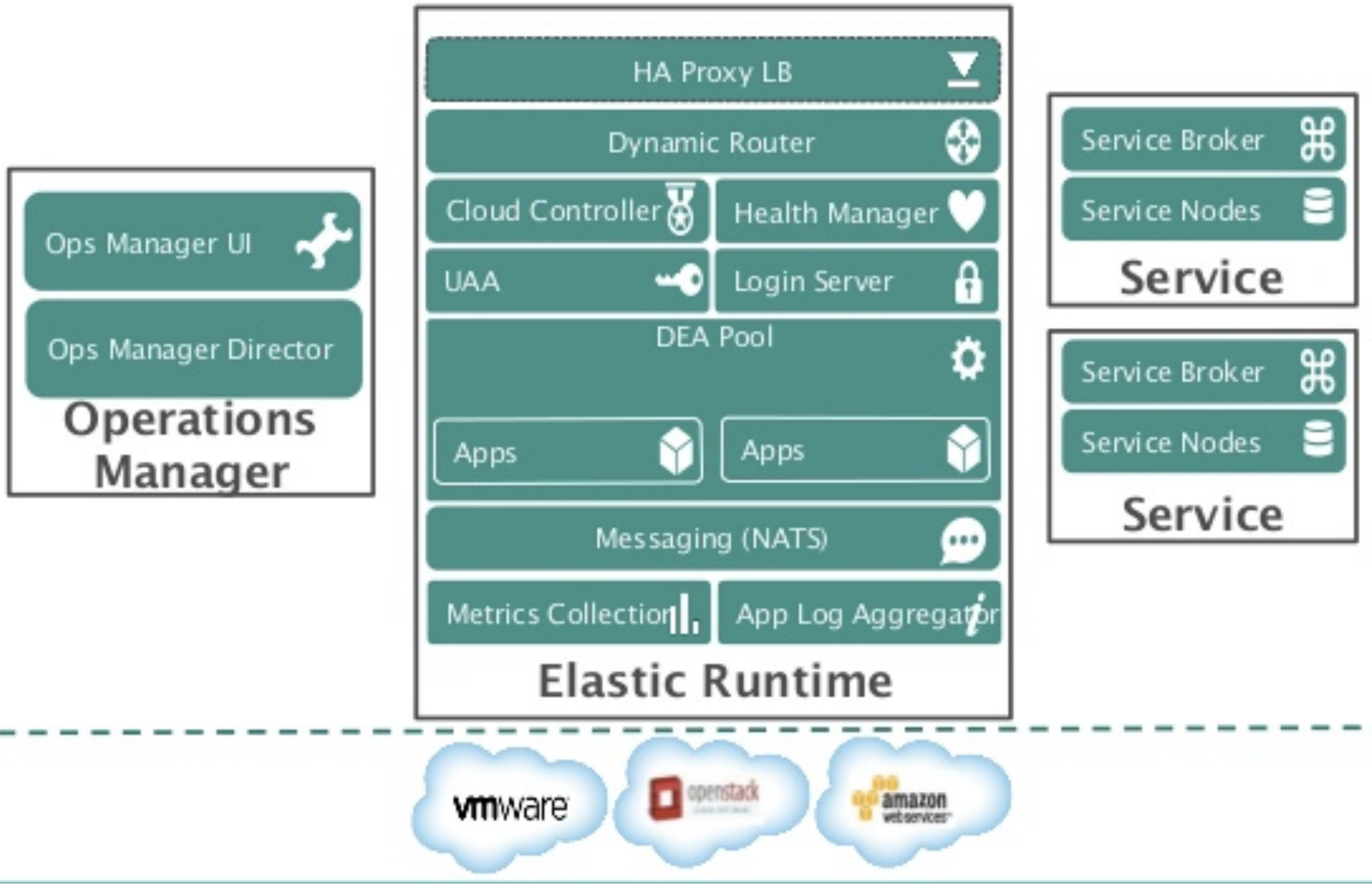
Email	Application Development	Caching
CRM	Decision Support	Legacy, File
Collaborative	Web	Networking, Technical
ERP	Streaming	Security, System Mgmt

CONSUME | **BUILD ON IT** | **MIGRATE TO IT**

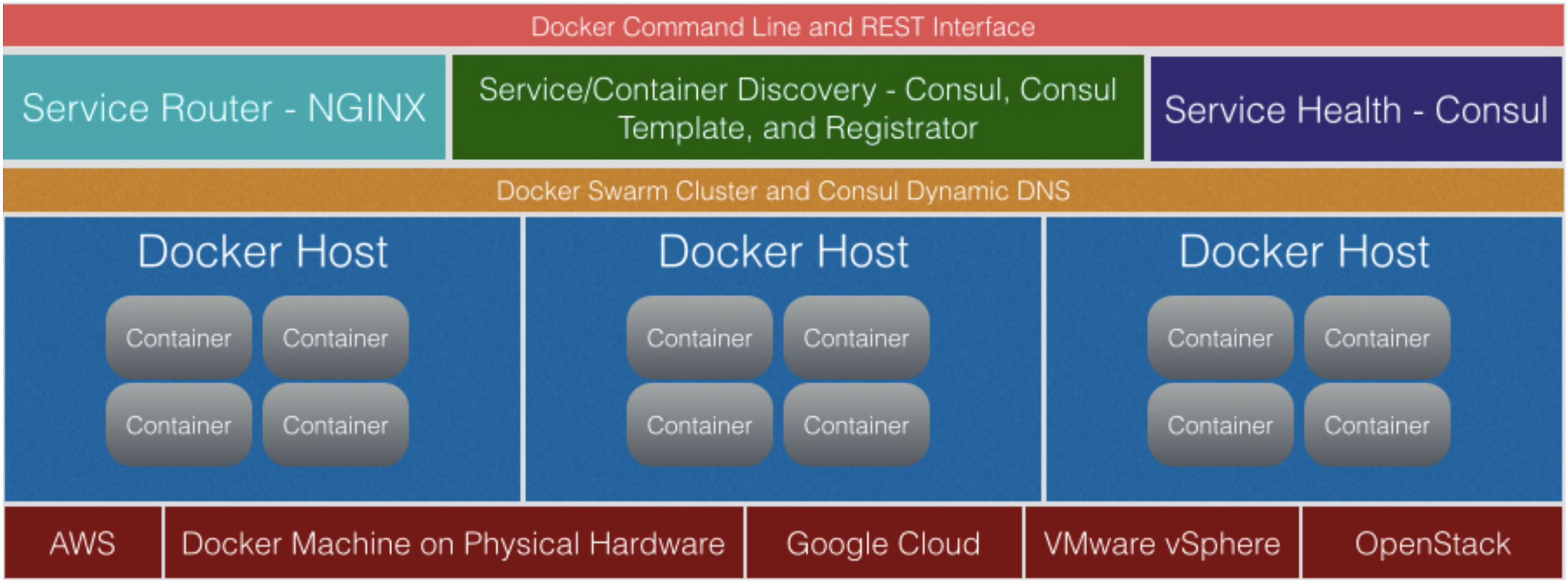
apprenda


Apprenda Platform
.NET • JAVA • TOMCAT • IIS • SQL SERVER • WEBSPHERE • WEBLOGIC • WCF • JBOSS • ORACLE
Your Standard OS, Virtual Machine, and Application Server
Application Centric Infrastructure (ACI)
CISCO
Unified Computing System (UCS)
PRIVATE
HYBRID
PUBLIC
Metapod
CISCO
Intercloud
apprenda
CISCO

Pivotal CF Architecture



Back to Docker….



Core Devops Tech

Common Elements of the Software Supply Chain

sonarqube

Nexus

maven

Jenkins

JIRA

git

puppet labs

GitLab

CHEF

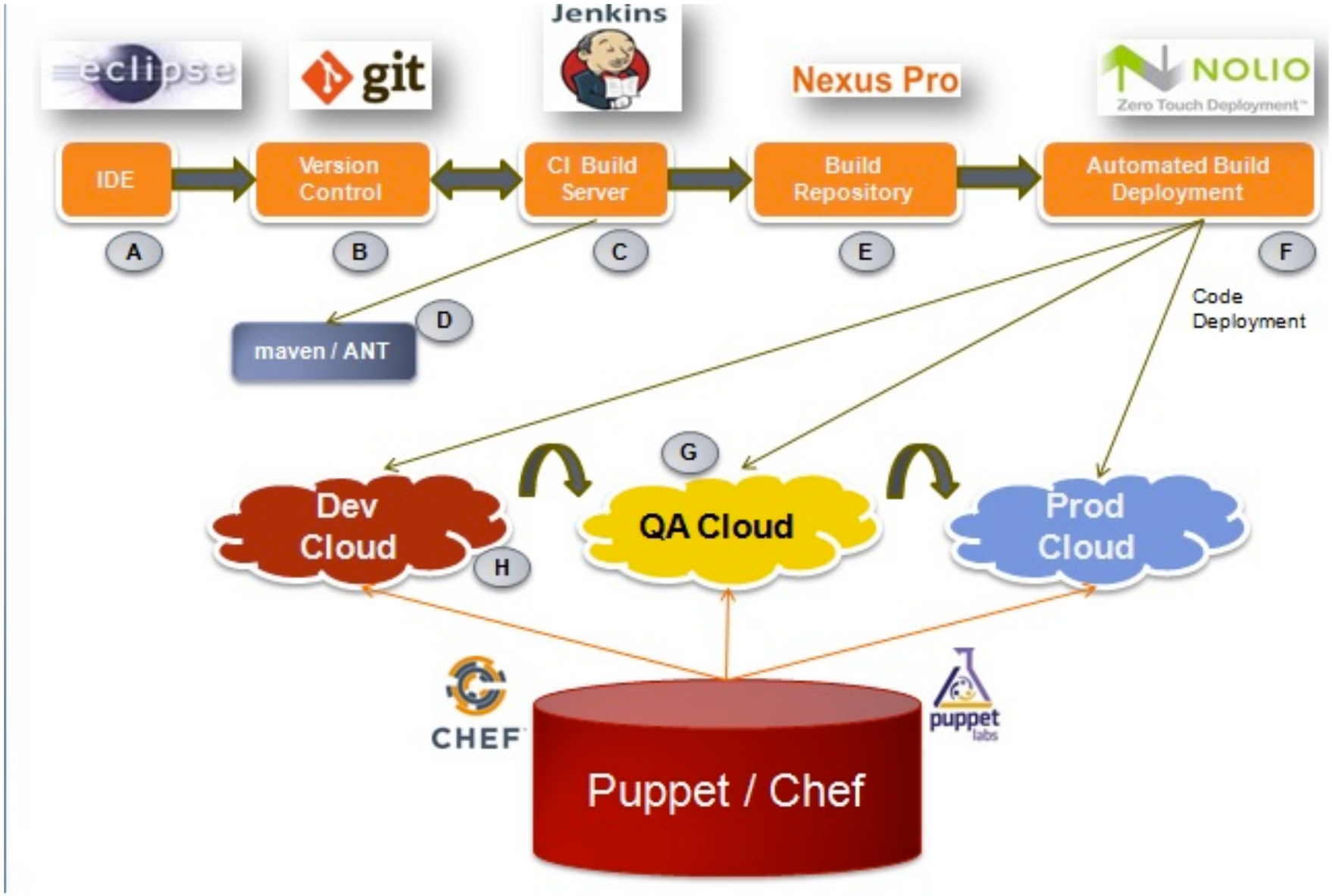
docker

Apache Tomcat

VAGRANT

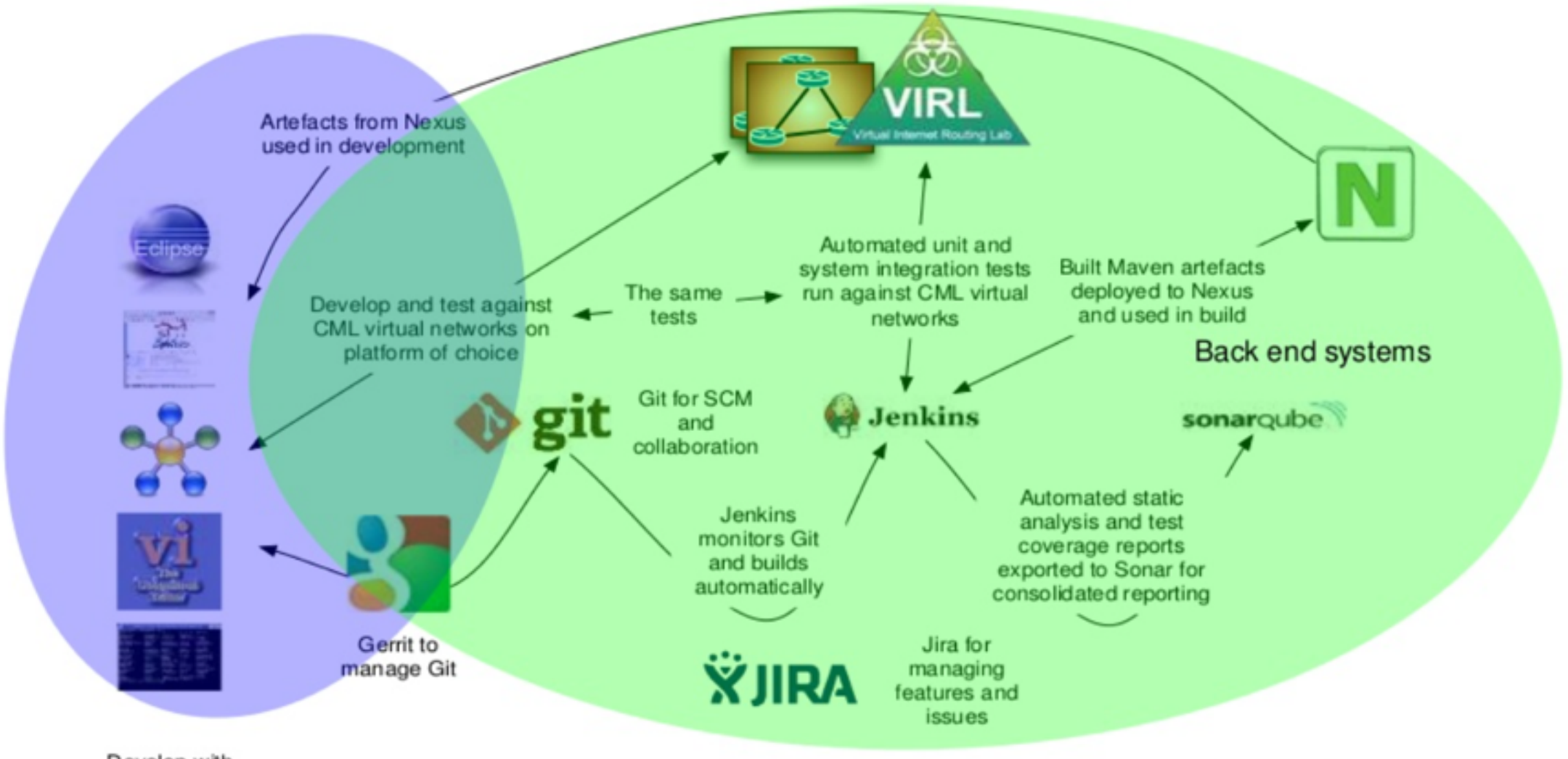
ANSIBLE

Example Reference Architecture for DevOps



Source: http://agiletrick.com/devops/architecture-of-devops/

According to Cisco



General Notes

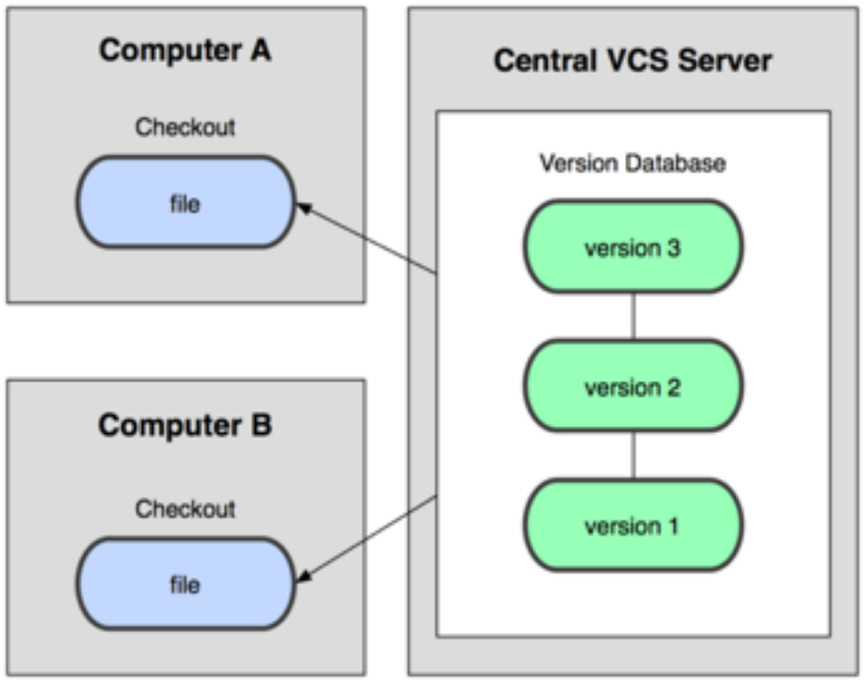
- CAPS (Chef, Ansible, Puppet, Salt) are mainly for centrally controlling what lives inside a large number of instances. I.e. processes, files, etc.
- Terraform and CloudFormation are mainly for creating instances themselves (and other cloud resources like load balancers etc).

Git

- Source Control
- The backbone for open source

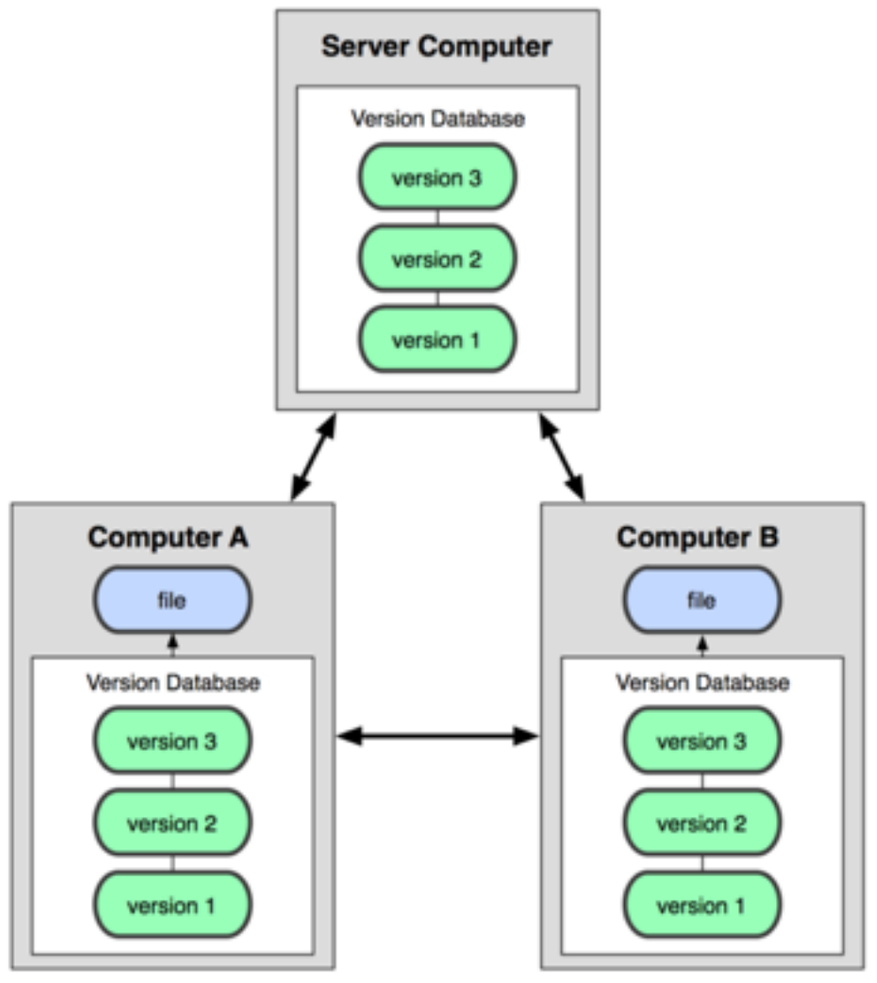
Git uses a distributed model

Centralized Model



(CVS, Subversion, Perforce)

Distributed Model

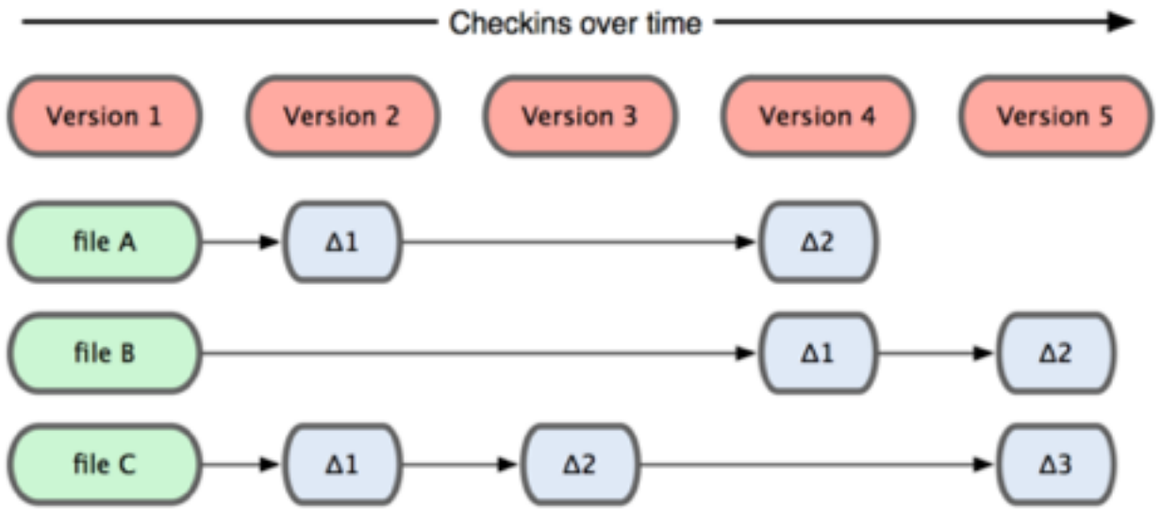


(Git, Mercurial)

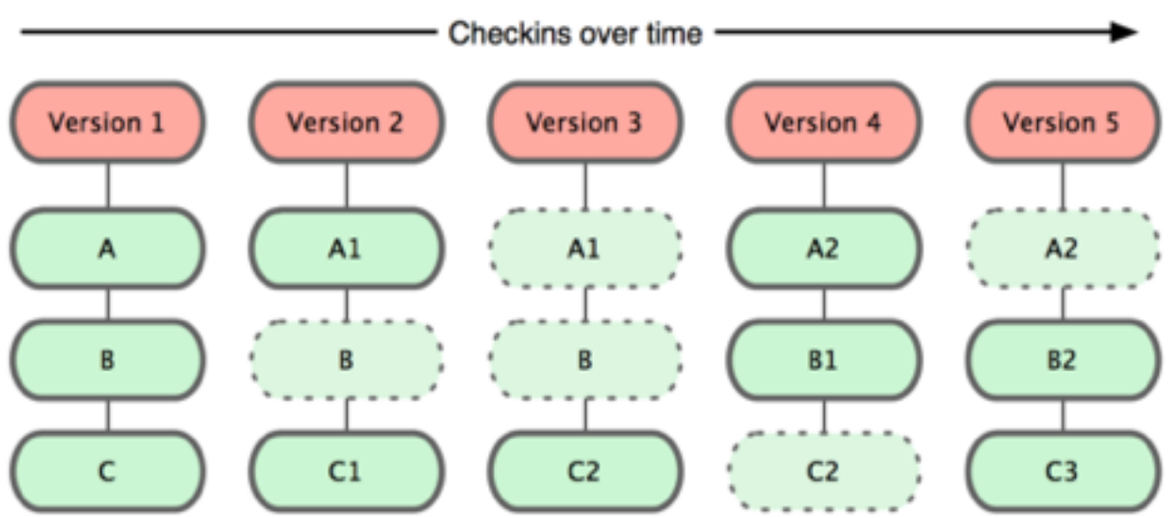
Result: Many operations are local

Git takes snapshots

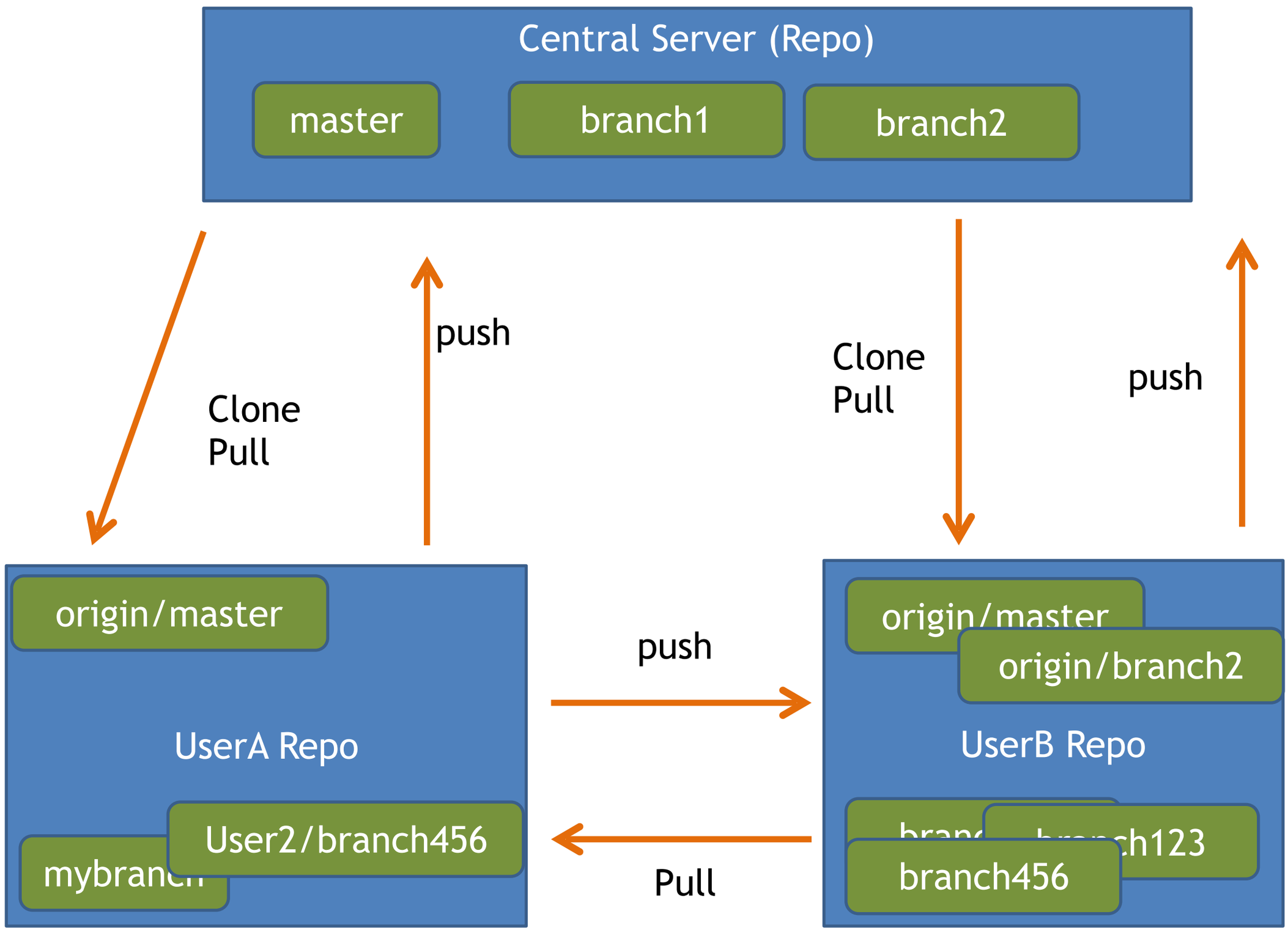
Subversion



Git



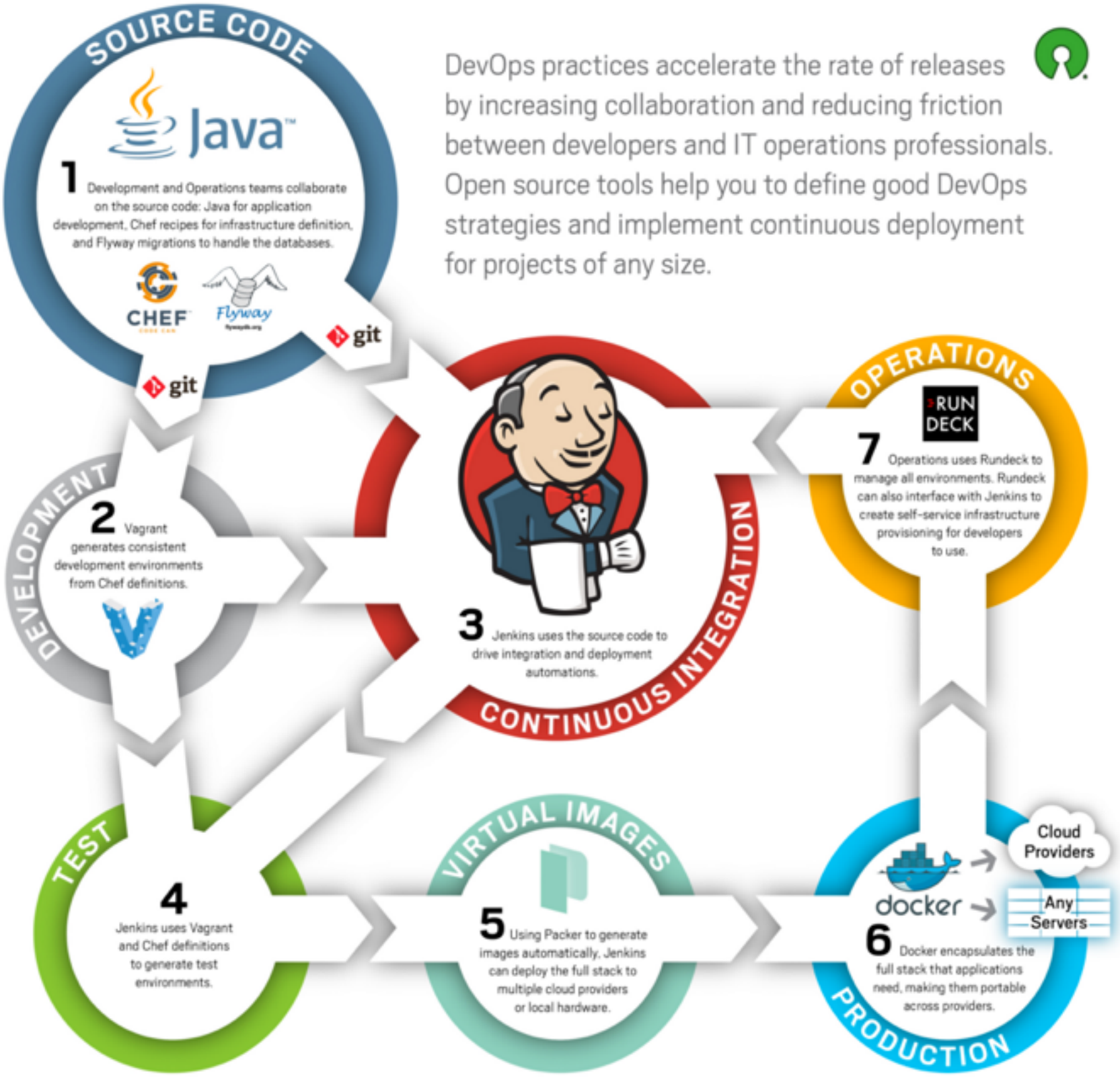
How Git Does it



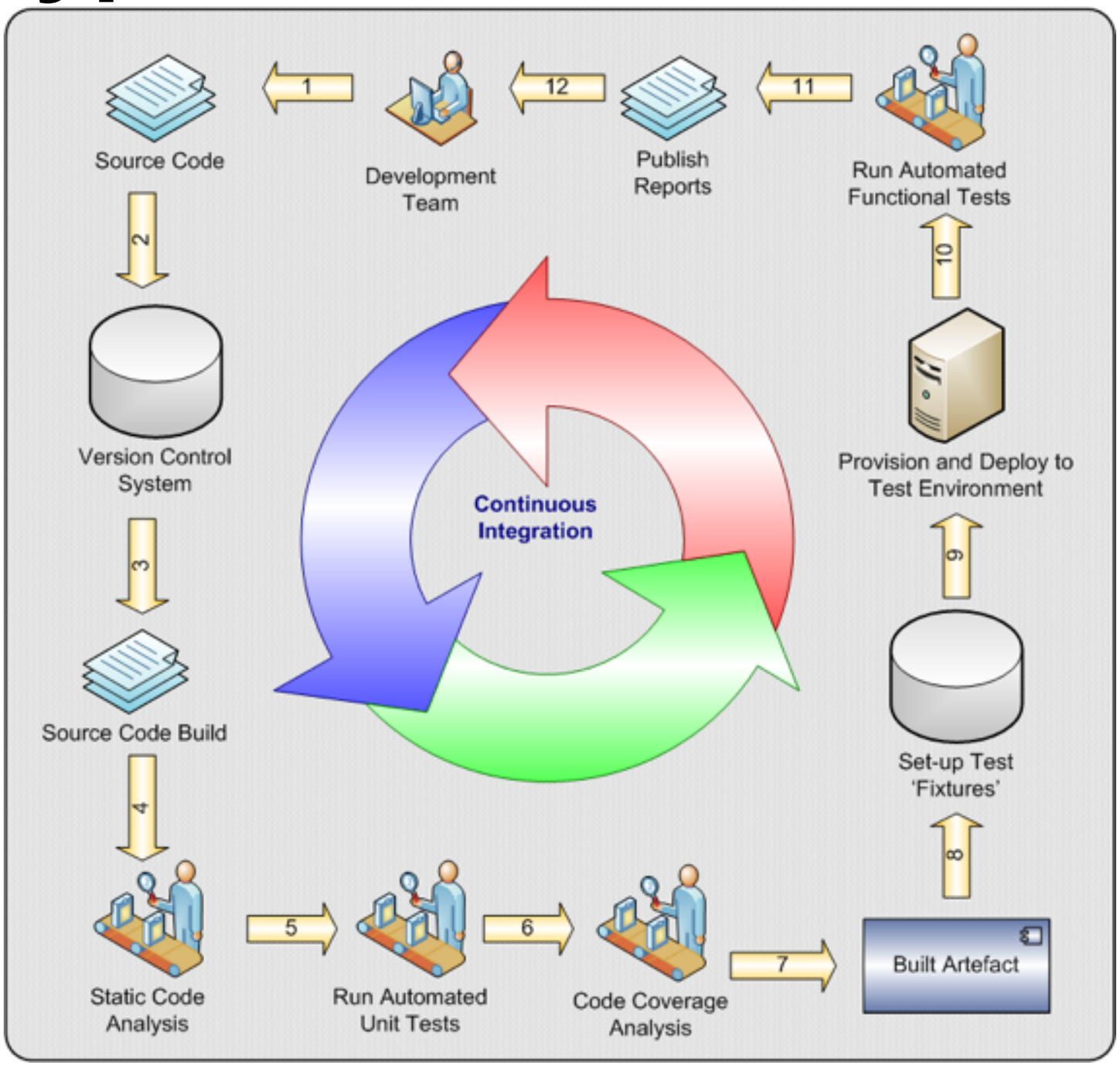
Jenkins

- Continuous integration server
  - Coordinate “jobs” to automate the building of applications and environments
  - Scales using master/slave architecture
  - Integrates into all major DevOps and code build tools

DevOps practices accelerate the rate of releases by increasing collaboration and reducing friction between developers and IT operations professionals. Open source tools help you to define good DevOps strategies and implement continuous deployment for projects of any size.



Typical CI Workflow using Jenkins



Jenkins Pipelines

Vagrant

- Provisioning Tool for Dev & Test Environments
- Why use Vagrant?
  - Quick
  - Easily replicate production on a Dev box
- Many use cases for developers, operations as well as QA

Vagrant Basics

- Command line utility
- Uses a vagrant file to describe the virtual environment you want to setup
- Uses a simple abstraction from underlying complexities
  - vagrant up
  - vagrant ssh
  - vagrant halt
  - etc.

Vagrant Basics

- The Box
  - Basic building block
  - Uses shared “repo”
- Providers
  - Supports AWS, VirtualBox, VMWare, etc

When

- Need to easily provision development environments

Pros

- Initial setup to get running is quick and easy
- Wide availability of boxes to provision your environment with

Cons

- Mysterious performance issues
- Some features on certain platforms (cough windows) can be a pain to implement

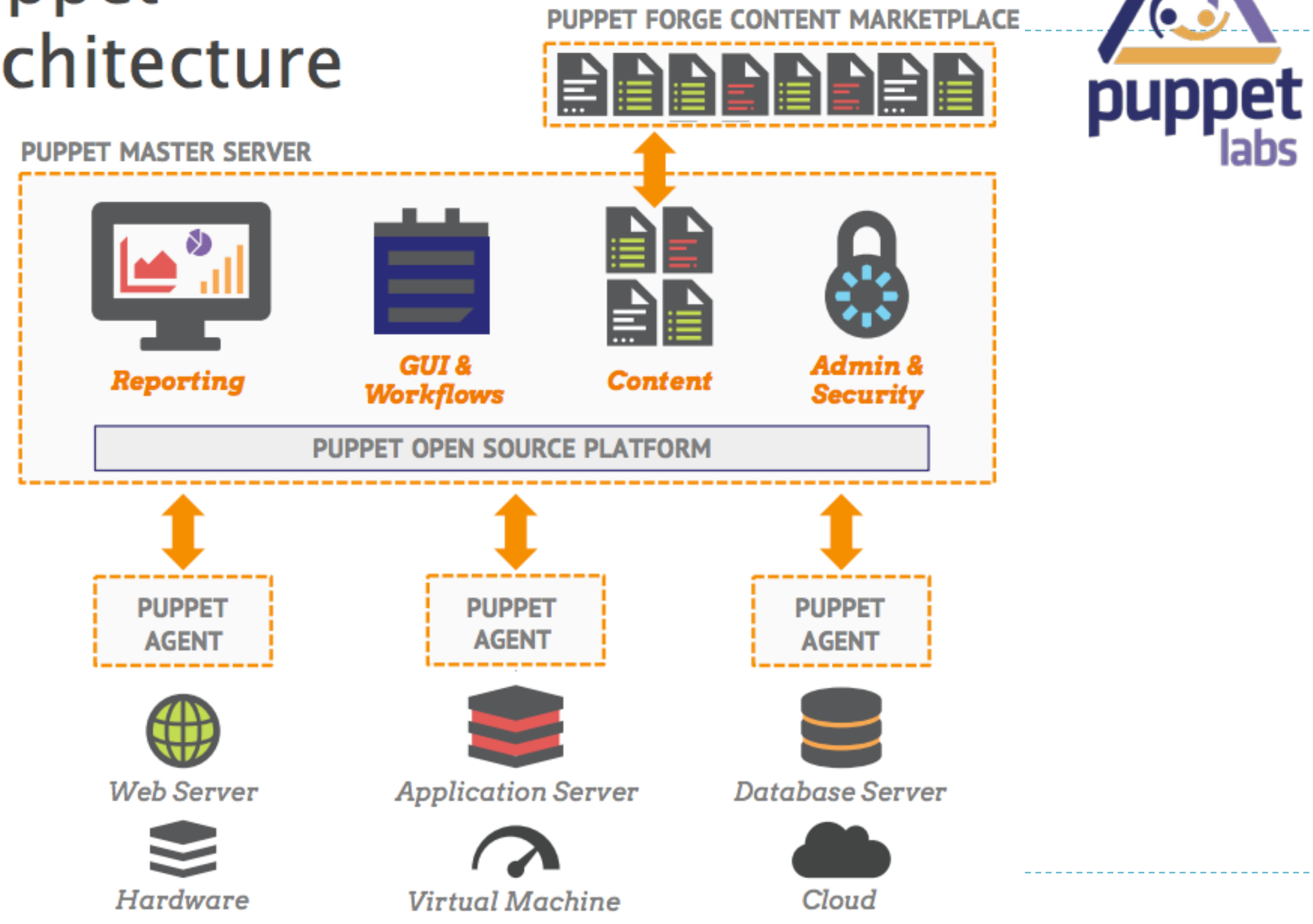
What is puppet?

- Think of it as a language.
- Describe state, not steps.
- Paint a picture of your ideal and most clean system.

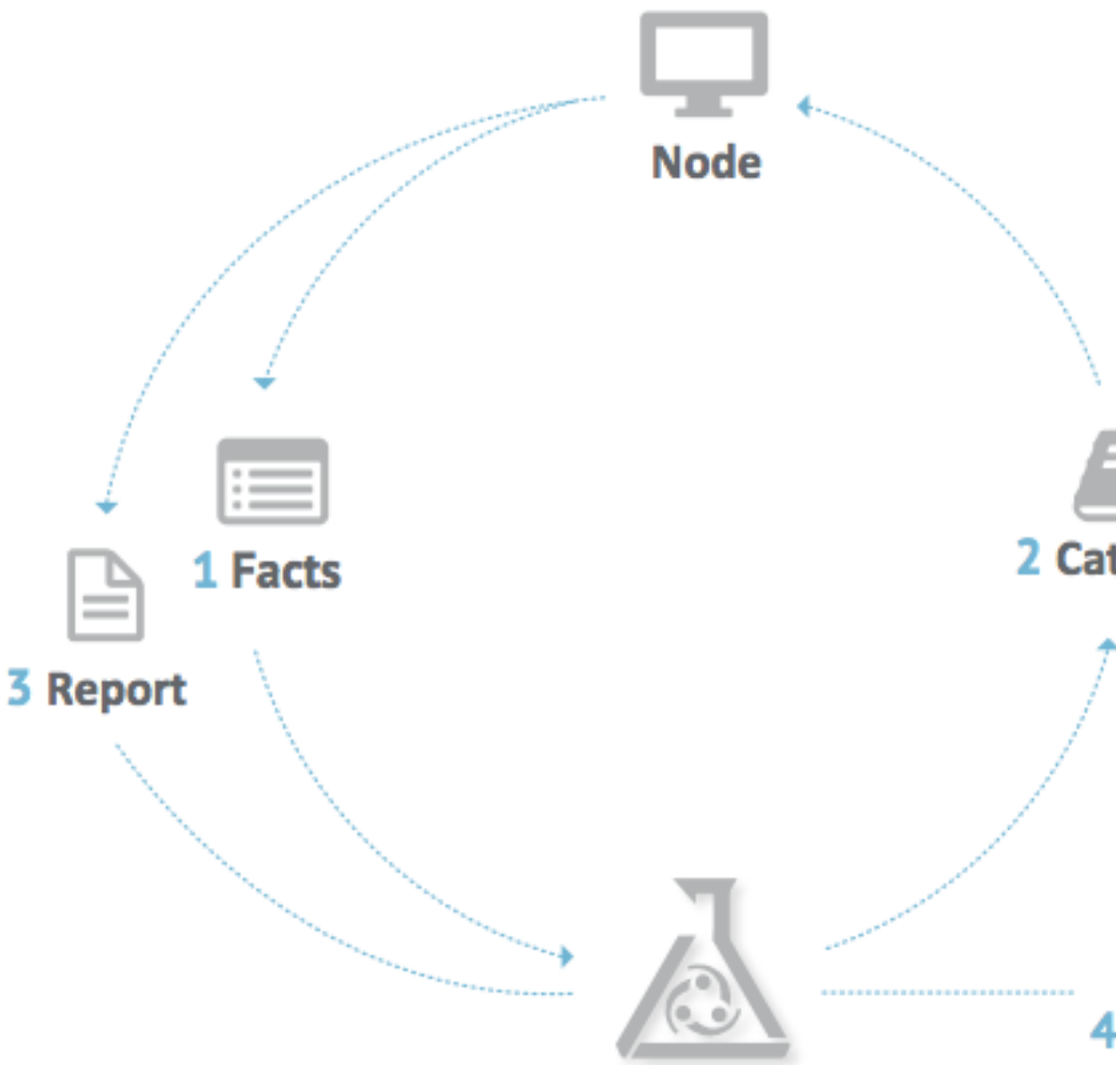
Puppet

- Automation of System Administration tasks
- Uses a declarative language to automate mundane admin tasks
- Seen as the tool of choice for pure configuration management
- Written in Ruby

Puppet Architecture



A Puppet Run



1. Facts
The node sends data about its state to the puppet master server.

2. Catalog
Puppet uses the facts to compile a Catalog that specifies how the node should be configured.

3. Report
Configuration changes are reported back to the Puppet Master.

4. Report
Puppet's open API can also send data to 3rd party tools.

When

- Stability and maturity are key factors
- Good for large organizations with a heterogeneous environment and breadth of skills on the team.

Pros

- Mature interface and runs on nearly every OS.
- Simple installation and setup.
- Complete Web UI in this space.
- Strong reporting capabilities.
- Well-established support community

Cons

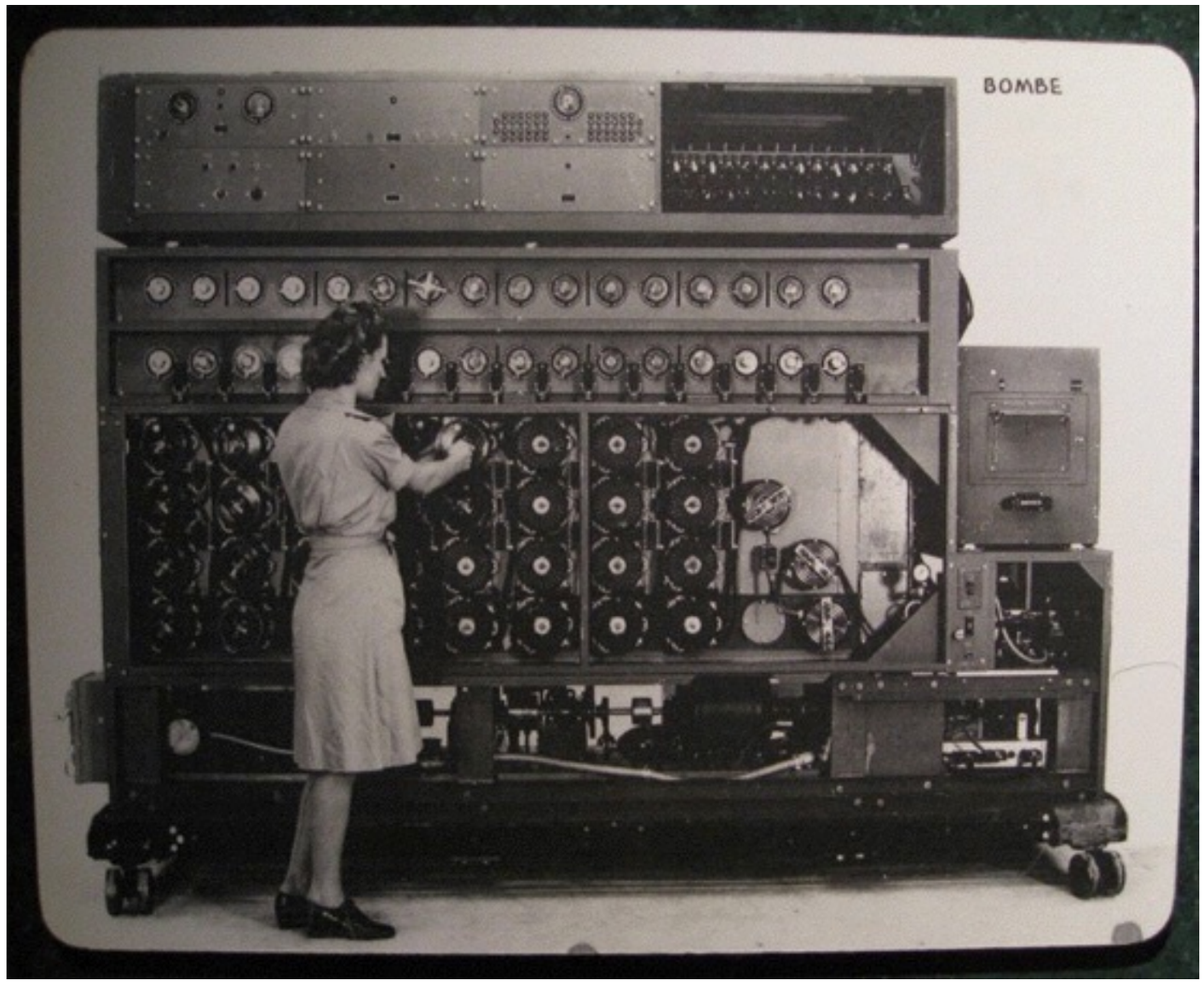
- Advanced tasks will lead you to use the CLI, which is Ruby-based
- Due to the DSL and a non simplistic design, a Puppet code base can grow large, complex, and will be hard for new team members to become productive on quickly.
- Pure-Ruby version support is being scaled back.
- Model-driven vs code driven approach

Chef

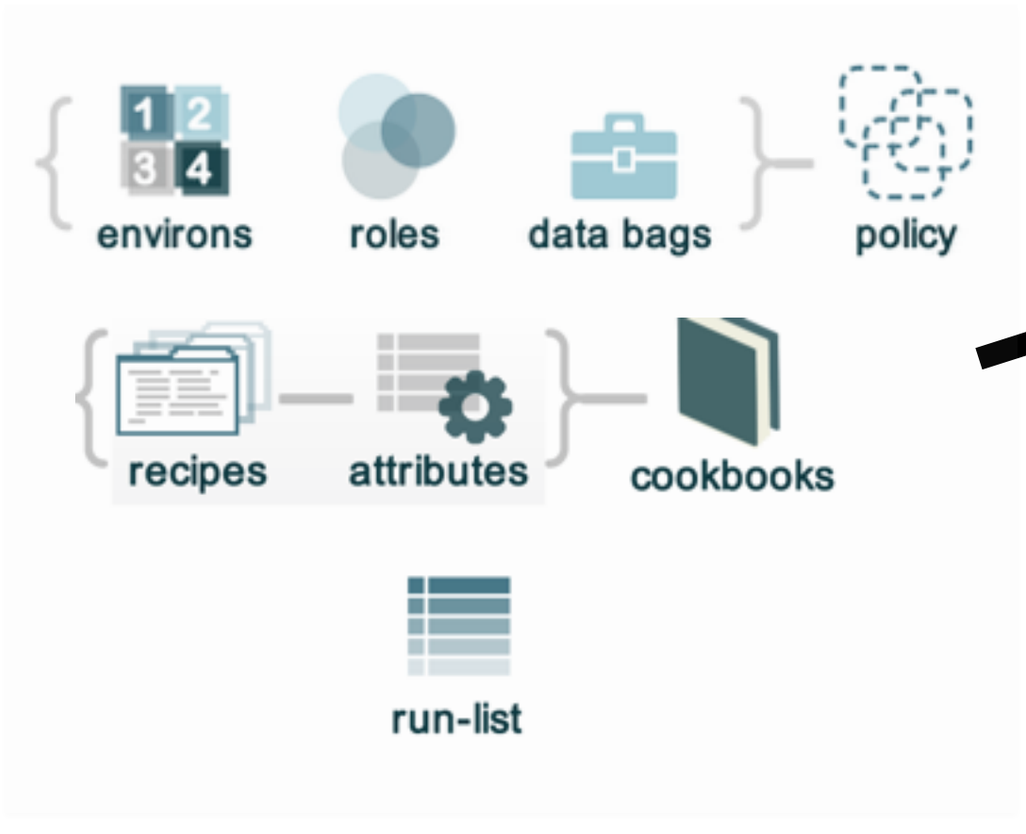
- Chef is the ninja of DevOps tools
- Comprehensive utility for
  - provisioning
  - managing
  - automating entire infrastructures
- Uses a client/server architecture to facilitate application of “recipes” to infrastructure components
- Written in Ruby

Chef Concepts

- Node
- Role
- Resource
- Recipe
- Cookbook
- Runlist



Run List



Enterprise Chef

What policy should I follow?

chef-client

run-list

"recipe[ntp::client]"
"recipe[users]"
"role[webserver]"

When/Why

- Organizations looking for a more mature solution for a heterogeneous environment.
- Good for development-focused teams.

Pros

- ‘Knife’ tool eases installation burdens.
- Large collection of modules and configuration recipes.
- Heavy focus on Git gives it strong version control capabilities.
- Code-driven approach gives you more control configurations.

Cons

- Not simple - can lead to large code bases and complex environments.
- Doesn’t support push functionality.
- Steep learning curve

Saltstack

- CLI-based tool that can be set up as a master-slave model or a non-centralized model.
- Python
- Push method of communication with clients
- Provides for grouping of clients and configuration templates

When

- Scalability and resiliency are a big concern.
- Oriented to system administrators due to its usability

Pros

- After setup, fairly straight forward
- DSL is robust with features
- I/O and configs are consistent-> YAML.
- Strong community.
- High scalability and resiliency in the master model
- Transparency is excellent. Easy to see what’s happening internally.

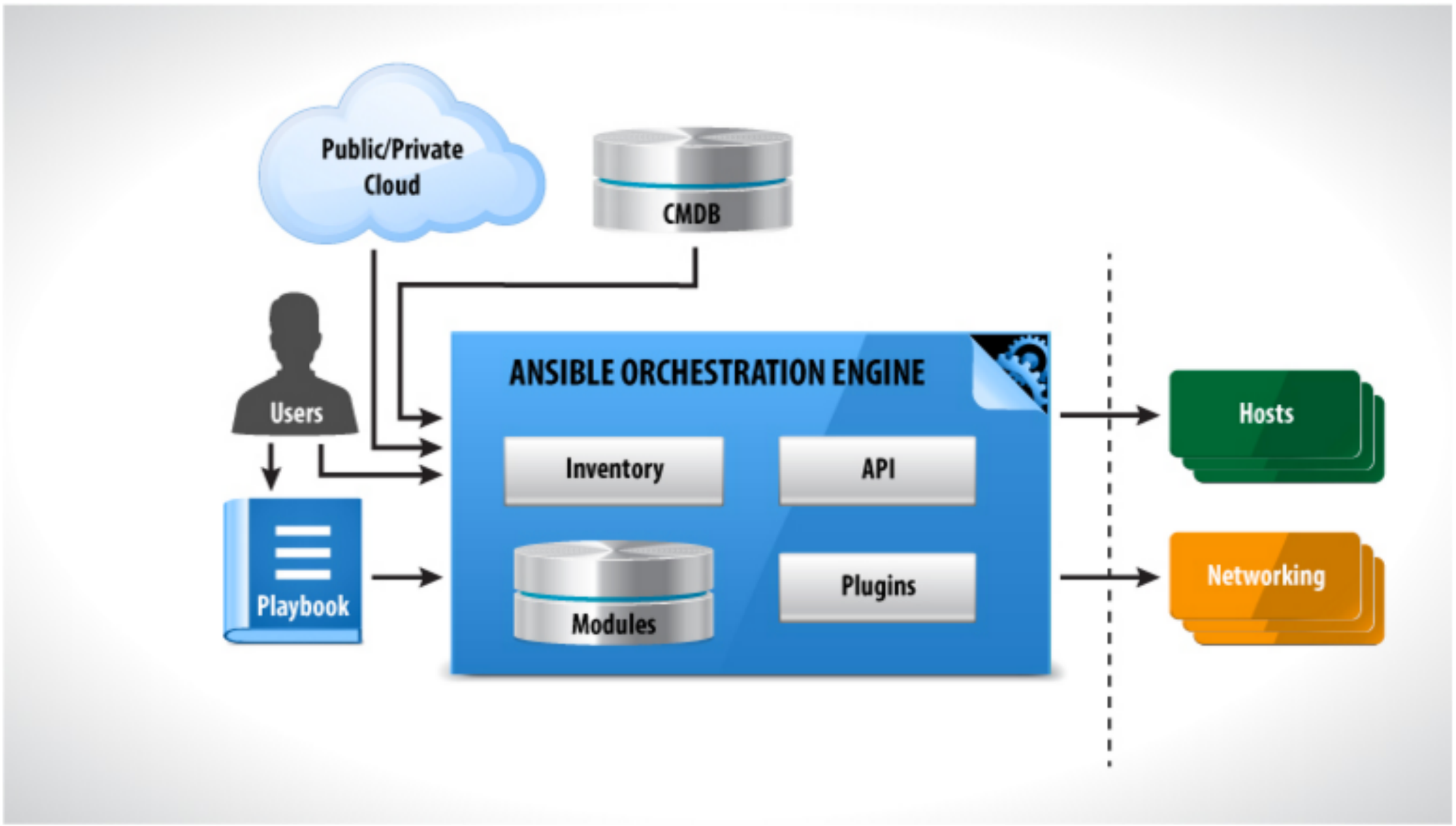
Cons

- Documentation is weak
- Incomplete web ui
- Non Lniux OS support weak
- Difficult to set up and to pick up for new users.

Ansible

- Agentless configuration management tool
- Written in Python
- Multi-node (like chef)
- Seeks to be simple
- Similar to Vagrant

Ansible Architecture



Ansible Key Concepts

- Module
- Playbook

When/Why

- Getting up and running quickly
- Ease is important to you
- You don’t want to install agents on remote nodes or managed servers
- It’s good if your focus is more on the system admin side.

Pros

- SSH-based, so it doesn't require installing any agents on remote nodes.
- Easy learning curve thanks to the use of YAML.
- Playbook structure is simple and clearly structured.
- Has a variable registration feature that enables tasks to register variables for later tasks
- Much more streamlined code base than some other tools

Cons

- Less powerful than tools based in other programming languages.
- Does its logic through its DSL, which means checking in on the documentation frequently until you learn it
- Variable registration is required for even basic functionality, which can make easier tasks more complicated
- Introspection is poor. Difficult to see the values of variables within the playbooks
- No consistency between formats of input, output, and config files
- Struggles with performance speed at times.

Teraform

- Higher level abstraction of CM
- Written in G (golang)
- Focuses on data center wide operations
- Integrates with other CM tools like Chef and Puppet
- From the makers of Vagrant

Terraform Configuration

- Like Vagrant
  - Command line tool
  - Uses text files to describe the infrastructure

When

- The planning phase plain and simple.
  - Killer feature
- When the holes in a young product are not in areas you need support. (Missing resources)

Pros

- Supports many cloud providers and can run on your local machine
- Vendor neutral
- Separate planning and execution phase - killer feature
- Young - stability can be an issue

Cons

- Limited support for many AWS components
- Issues with state management - state file sharing between developers is chaos
- DSL is young and missing some functionality

Cloud foundry

- Provision operating systems and middleware.
- Manage network security safe guards to prevent security threats.
- Manage application connections to external sources including databases and legacy middleware.
- Provides 4 levels of HA, as well as built in load balancing for scale
- Supports multi-tenant environments - each line of business can operate with a discrete quota and isolated system access.
- Provision next generation data services including NOSQL databases, traditional databases and hadoop clusters.
- We provide horizontal and vertical scaling for the underlying IaaS so that you can scale your infrastructure in lock step with your Business.
- Provides a built-in log aggregation service

Open Source Cloud Foundry Capabilities

Automatic AppServer & OS Configuration with Buildpacks *("just push your app")*	Application Containerization & Cluster Scheduling	Application Network Security Groups
Application to Services Binding and Access	App Health Mng, Load Balancing, Rapid Scaling, Availability Zones	Policy, Identity and Roles Management
Native & Extended Data, Mobile and Platform Services	IaaS Provisioning, Scaling & Configuration	Basic logging as a service

Deploy, Operate, Update & Scale with minimal downtime

vmware

amazon webservices™

openstack™

Cloud Foundry Components

BOSH CLI
CF CLI

HA Proxy LB

Dynamic Router

Cloud Controller

Health Manager

UAA

Login Server

DEA Pool

Apps

Apps

Messaging (NATS)

Metrics Collector

Log Aggregator

Elastic Runtime

Service Broker

Service Nodes

Service

Service Broker

Service Nodes

Service

When

- You are running on AWS
- Significant number of ready to go resource definitions

Pros

- Support for all AWS components
- No state management issues
- Templates written in son and very easy to understand
- Very stable

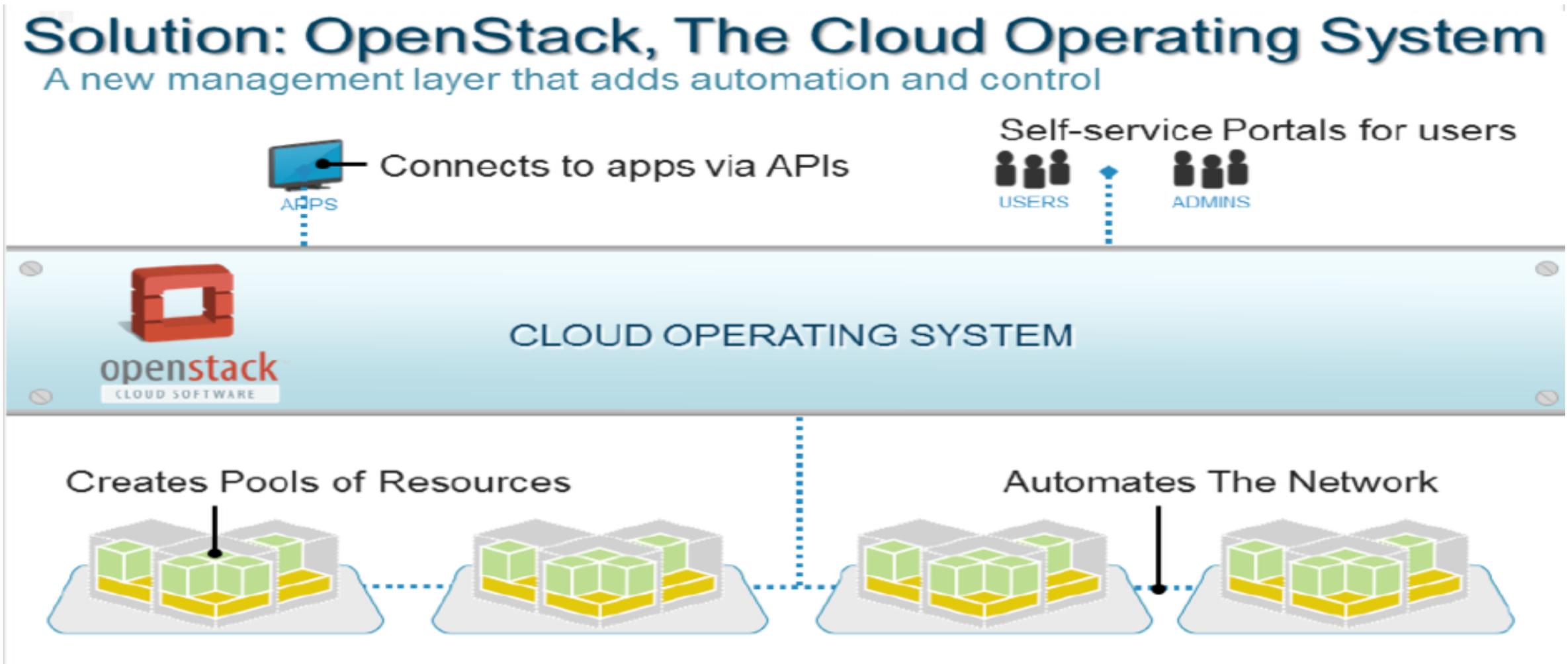
Cons

- AWS only and is a hosted service
- Can makes changes without prior notice - lack of transparency
- Issues with rollback mechanism

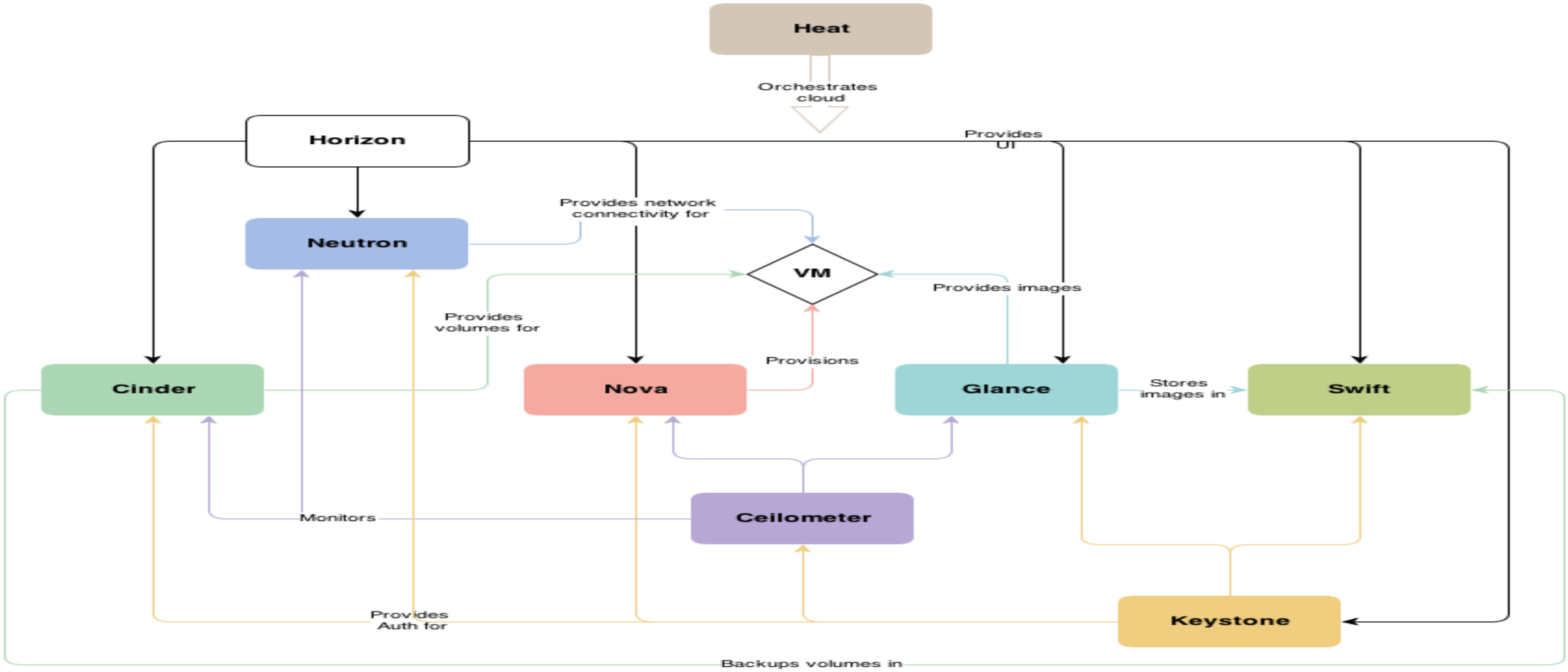
OpenStack

- Software platform for cloud computing
- Infrastructure-as-a-service (IaaS)
- The software platform consists of interrelated components that control hardware pools of processing, storage, and networking resources throughout a data center.

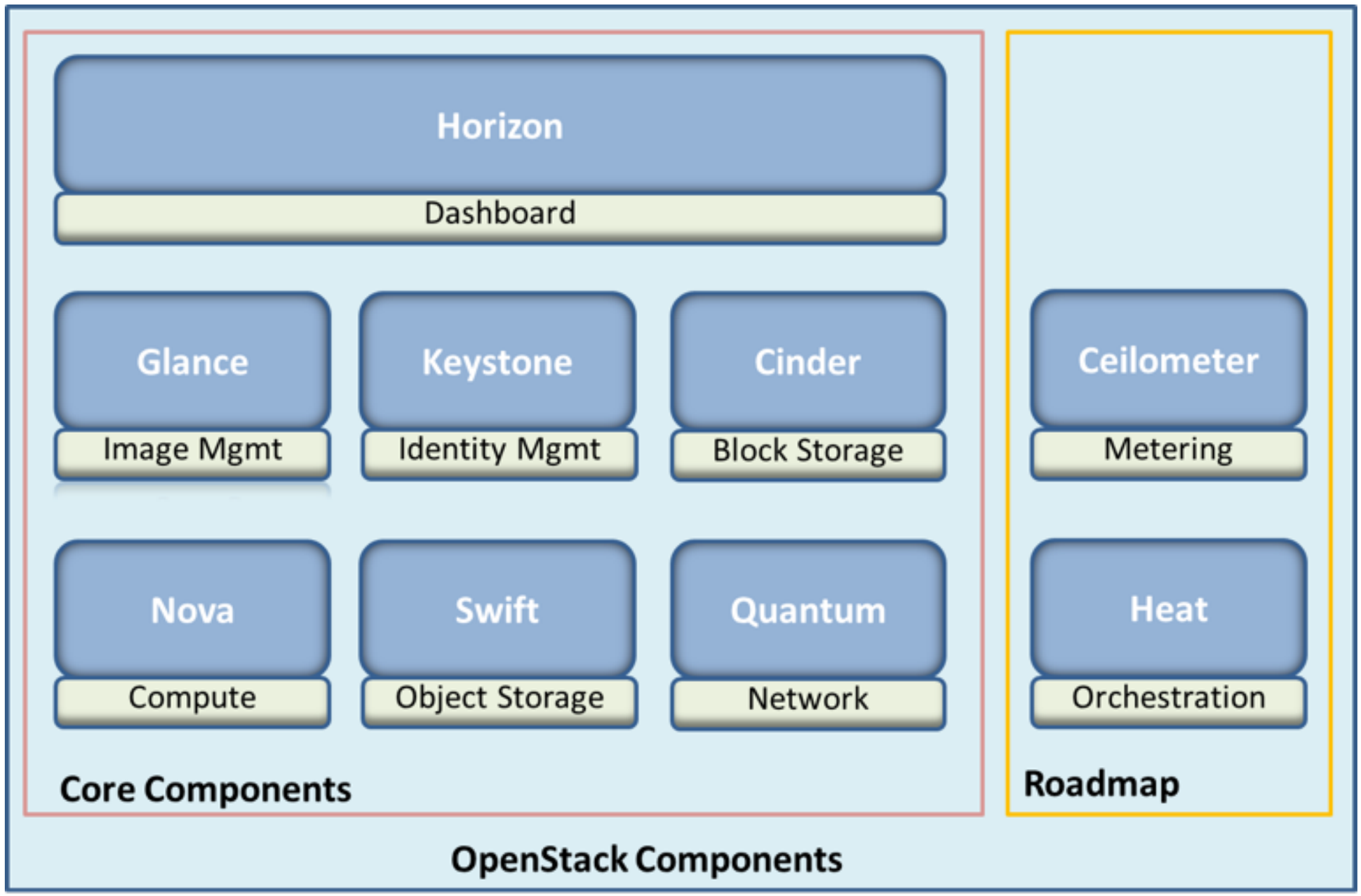
Automation and Orchestration of IT Resources



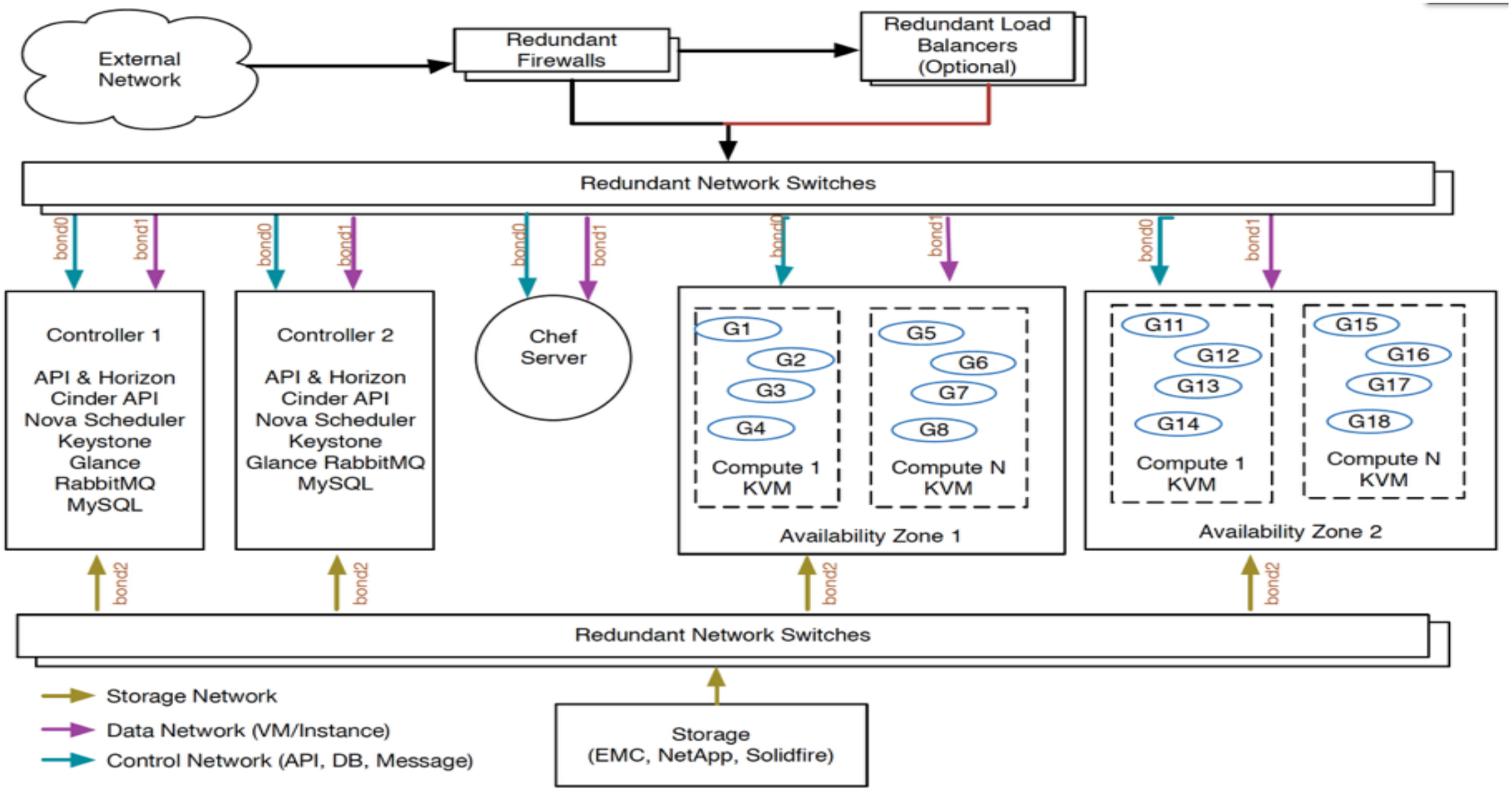
In a Loosely Coupled Architecture



By Leveraging Various Open Source Projects



Rackspace Private Cloud Reference Architecture



Discussion: Tool Diversity

Thank you!

For more information about our courses and services: