
WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

CPEN 321

Continuous Integration, DevOps, Services, Deployment

Part of the slides borrowed from the authors of "*DevOps: A Software Architect's Perspective*", L. Bass, I. Weber, L. Zhu, Addison-Wesley Professional, 2015.

Agenda

- **CI/CD**
- DevOps Overview
- Continuous Deployment (on the Cloud)
- Infrastructure as Code
- Monitoring

The story thus far

So far we covered

1. Gathering requirements from customers
2. Design
3. Choosing the best development strategy
4. Developing test plans

Still missing some some components:

- Automation
- Delivery
- Deployment

Ad-Hoc Integration

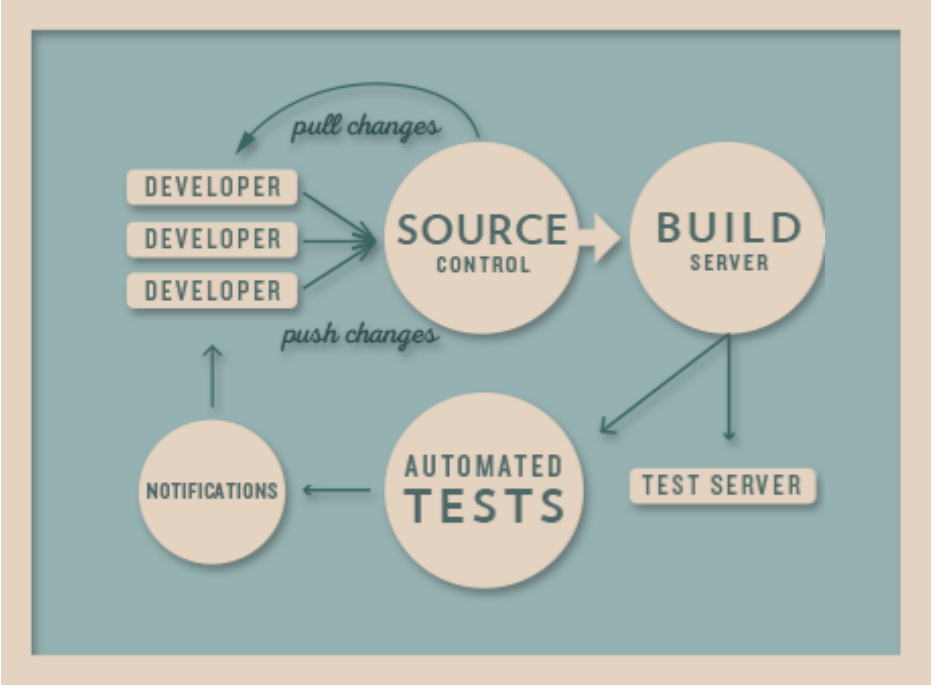
- Developers work in isolation on their own branches
- Often for weeks/months at a time
- Massive merges
- Difficult to test changes
- Difficult to verify state of branches

Leads to slow and painful release processes or "integration hell"

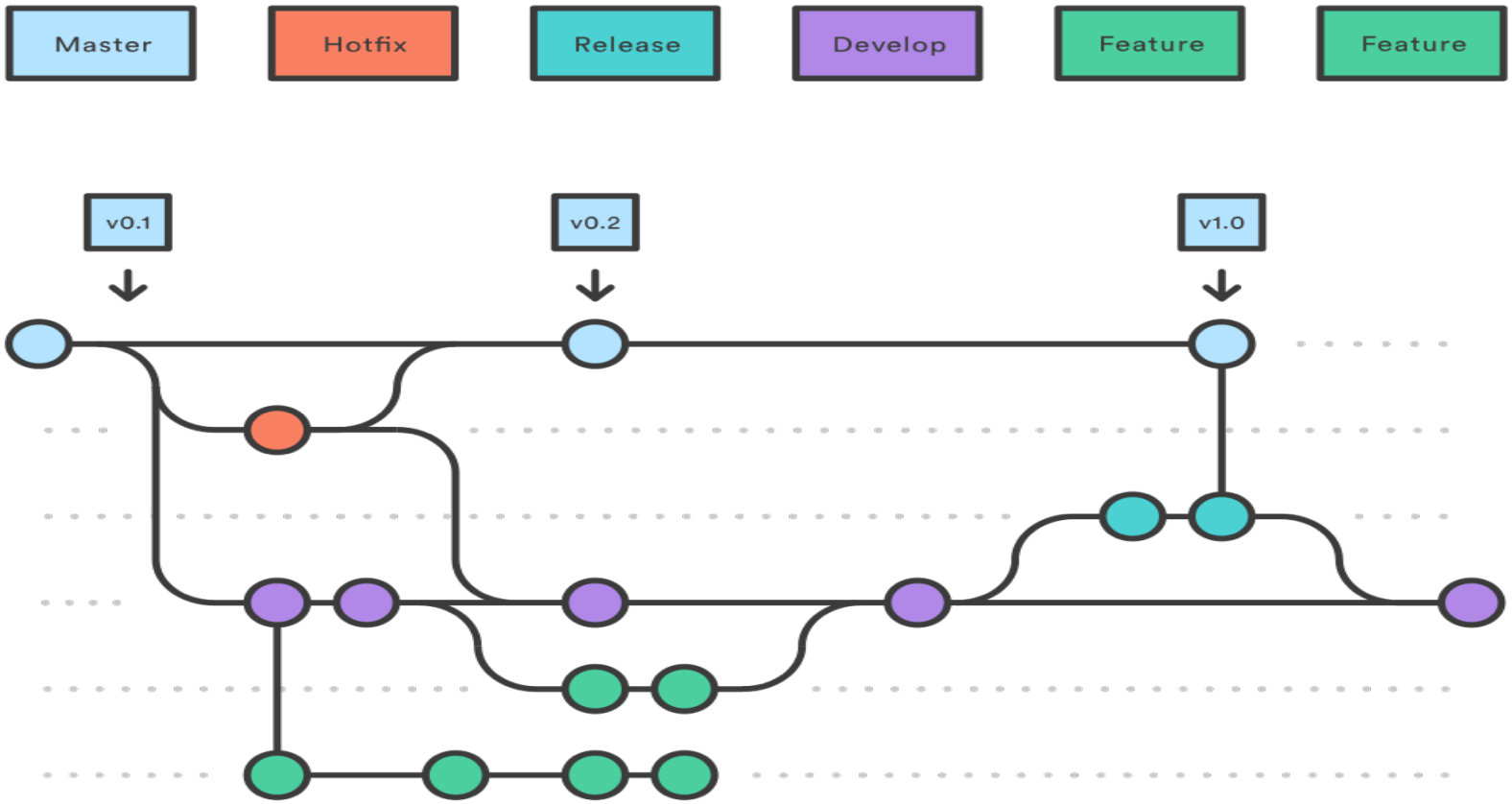
Continuous Integration

"Continuous integration is the practice of routinely integrating code changes into a main branch of a repository, and testing the changes, as early and often as possible." - Atlassian

It's a series of best practices! So just using tools will only help so much.

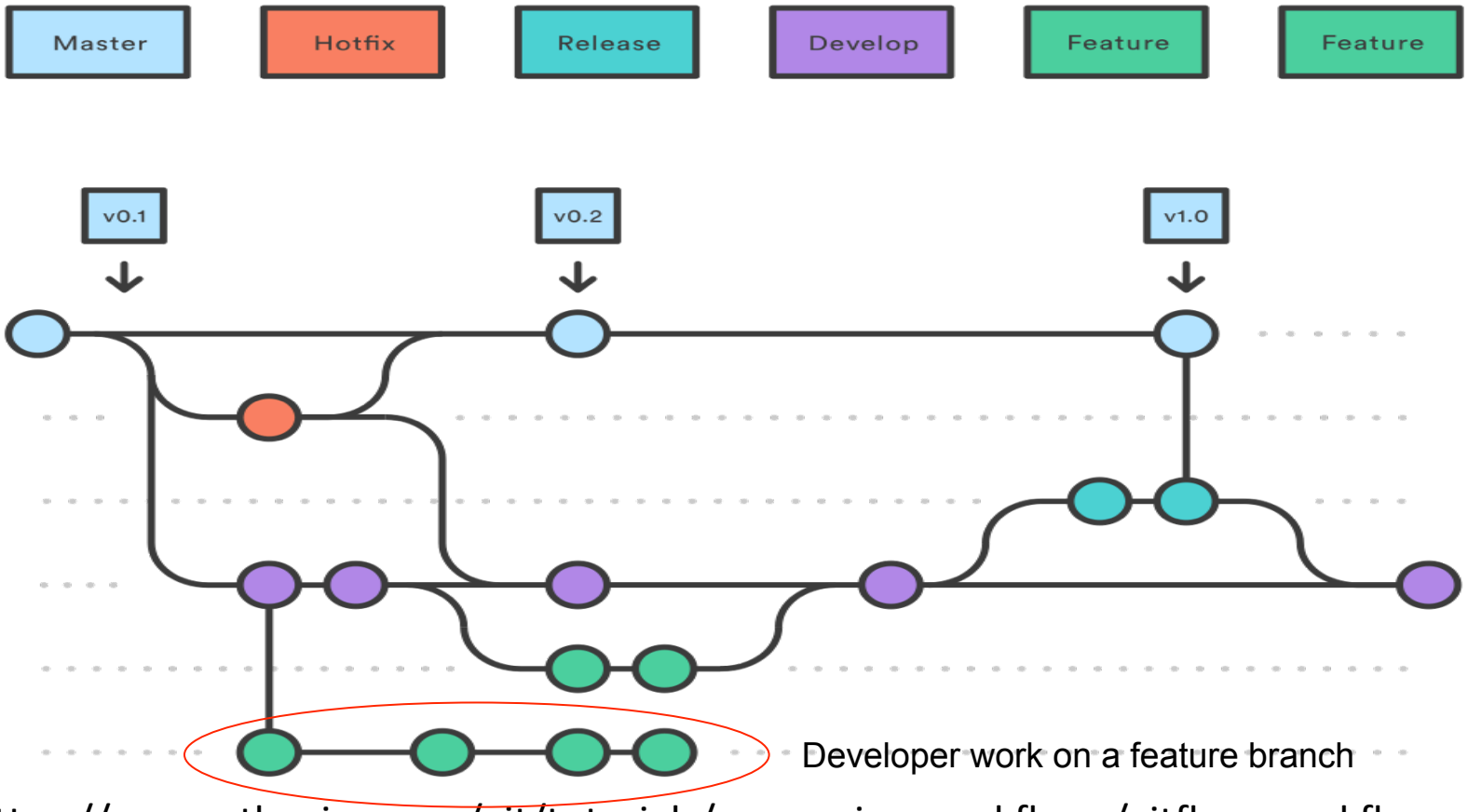


Continuous Integration - in practice



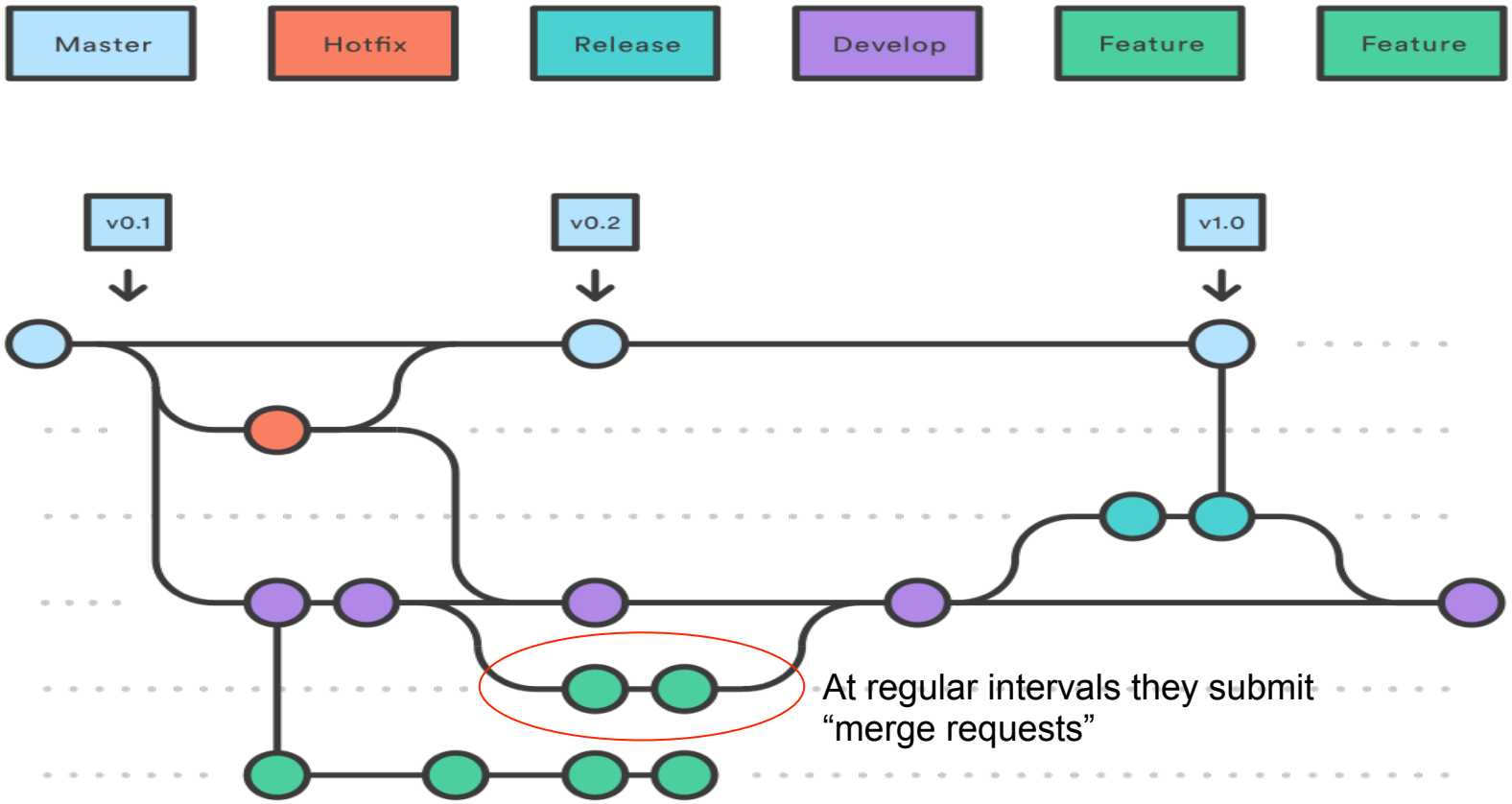
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Continuous Integration - in practice



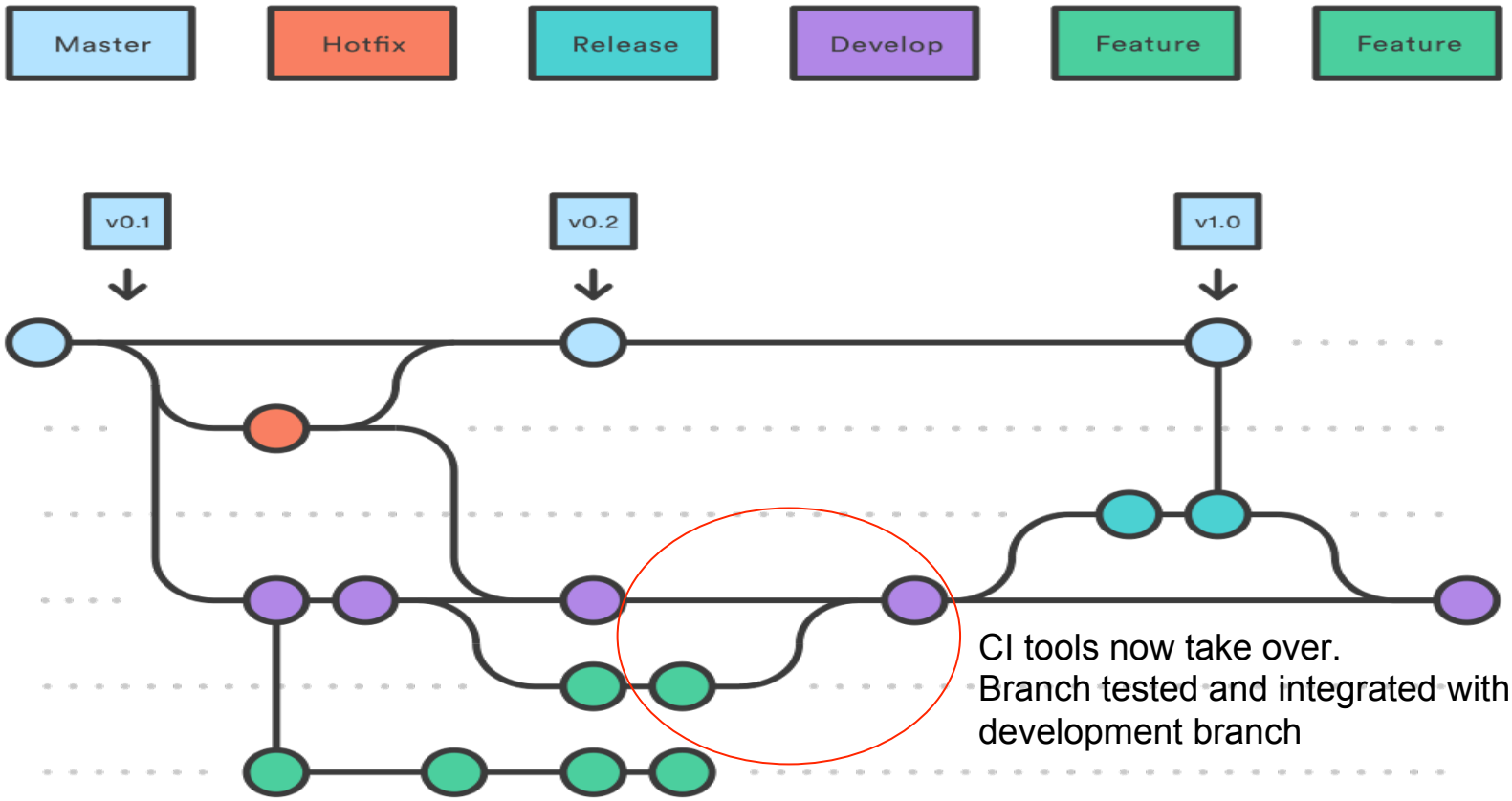
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Continuous Integration - in practice



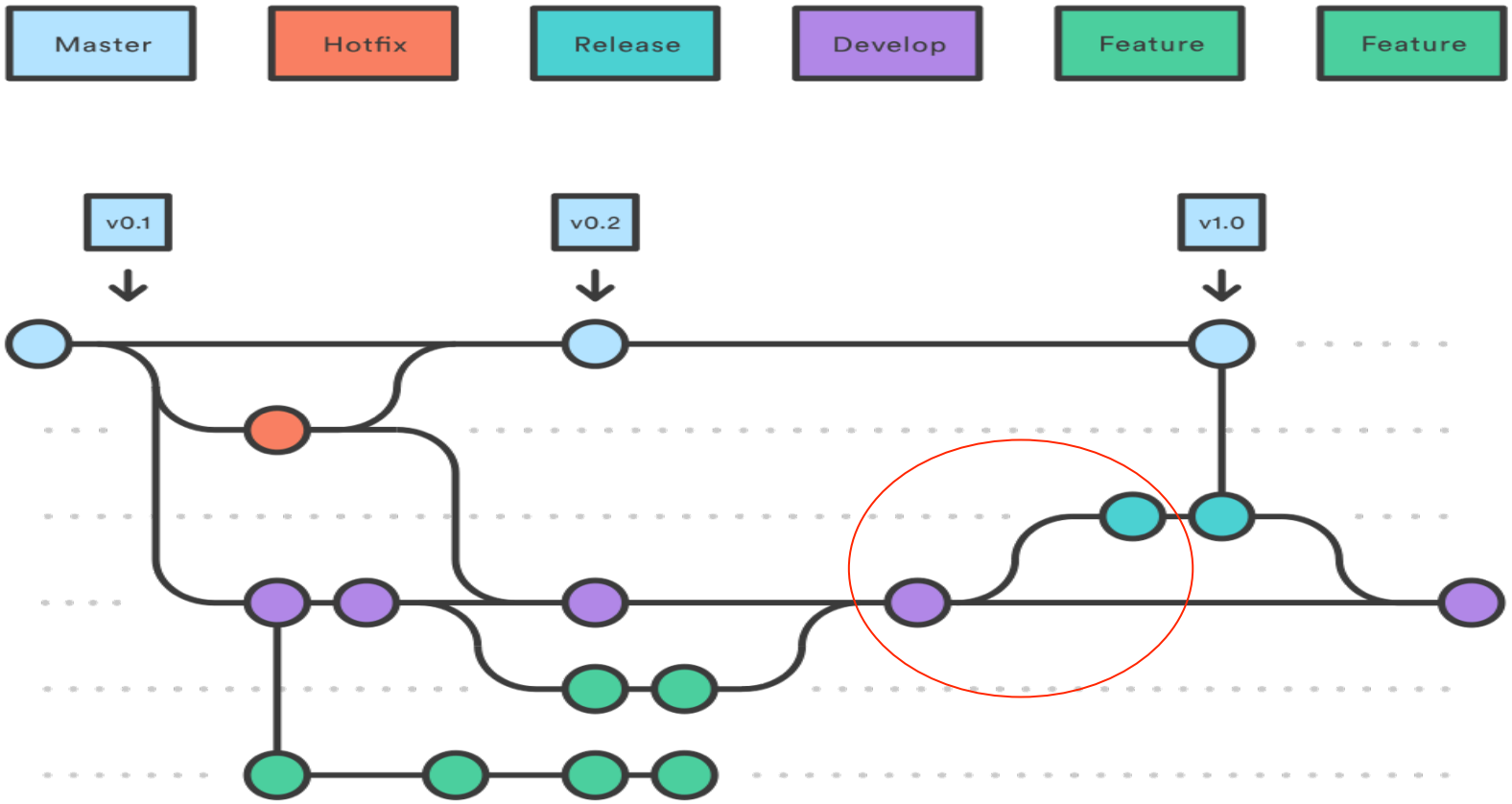
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Continuous Integration - in practice



https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Continuous Integration - in practice



https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Continuous Integration - tools

Many tools out there:

- Jenkins
- Travis
- Pipelines
- etc…

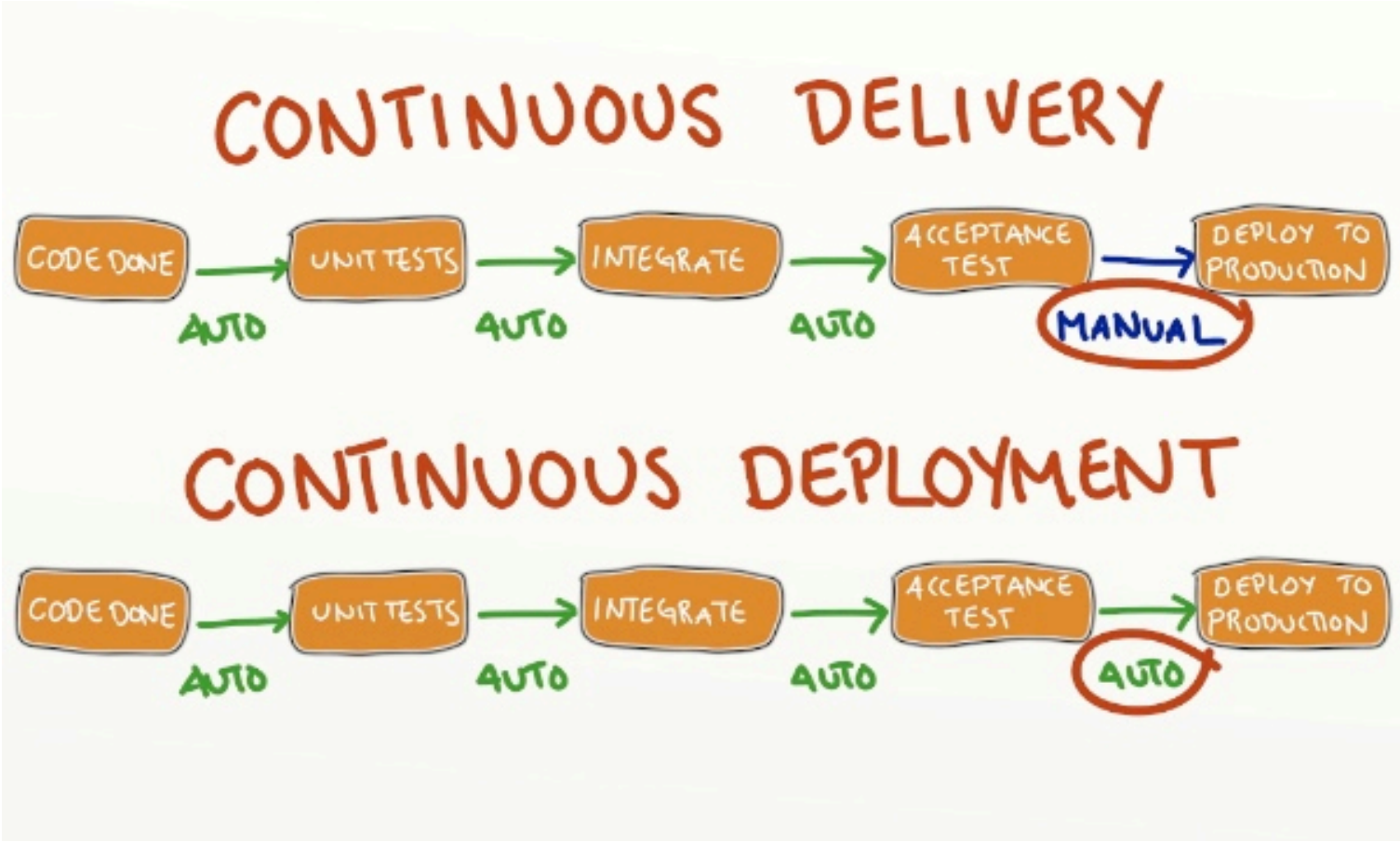
Most of these tools integrate with your revision control system and platform of choice.

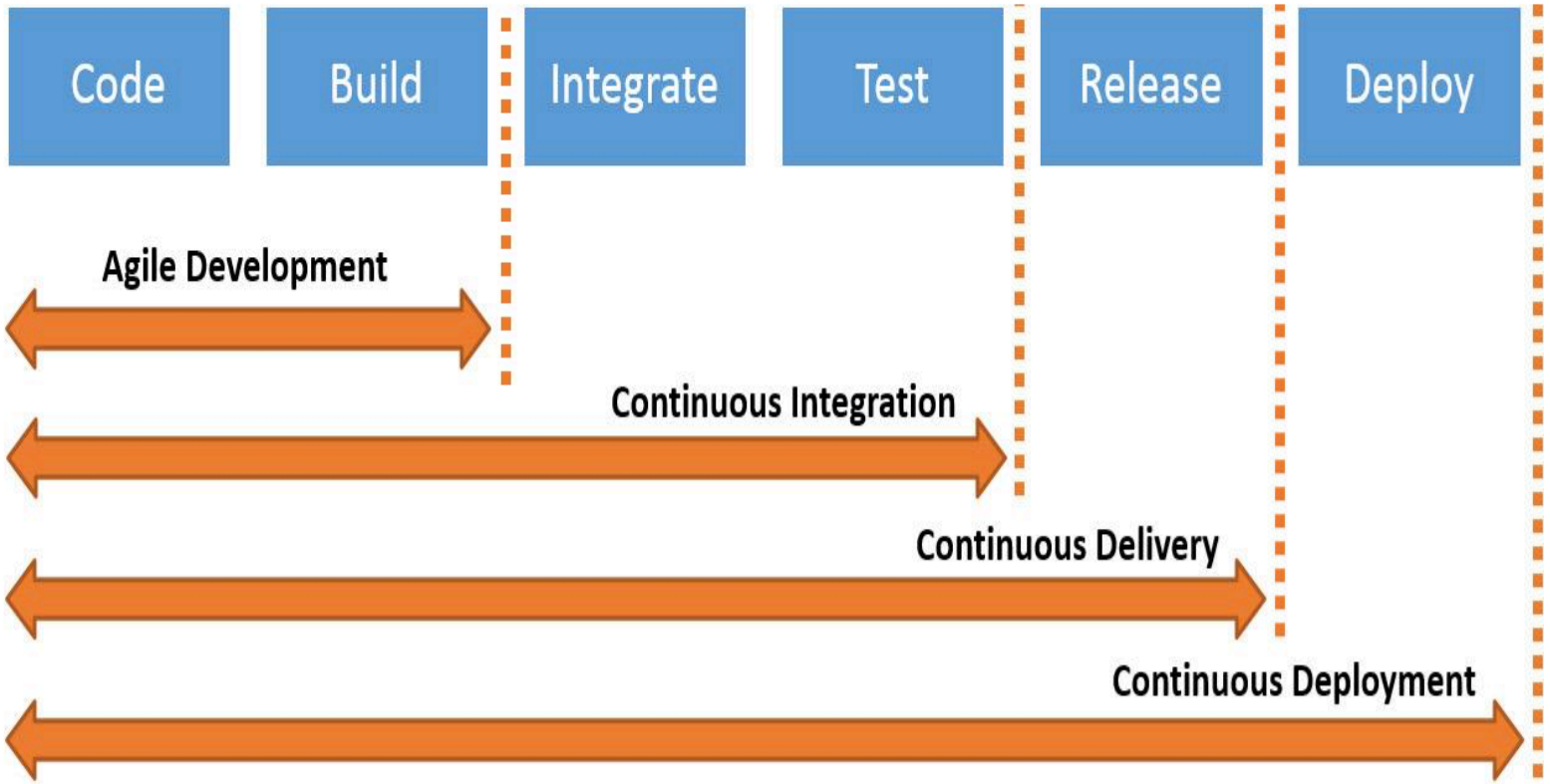
Some platforms even have CI built in!

What now?

There are only two steps to go:

1. Package/Deliver the product
2. Deploy onto the infrastructure


CONTINUOUS DELIVERY
CODE DONE
UNIT TESTS
INTEGRATE
ACCEPTANCE TEST
DEPLOY TO PRODUCTION
AUTO
AUTO
AUTO
MANUAL
CONTINUOUS DEPLOYMENT
CODE DONE
UNIT TESTS
INTEGRATE
ACCEPTANCE TEST
DEPLOY TO PRODUCTION
AUTO
AUTO
AUTO
AUTO



http://www.bmc.com/blogs/continuous-delivery-continuous-deployment-continuous-integration-whats-difference/

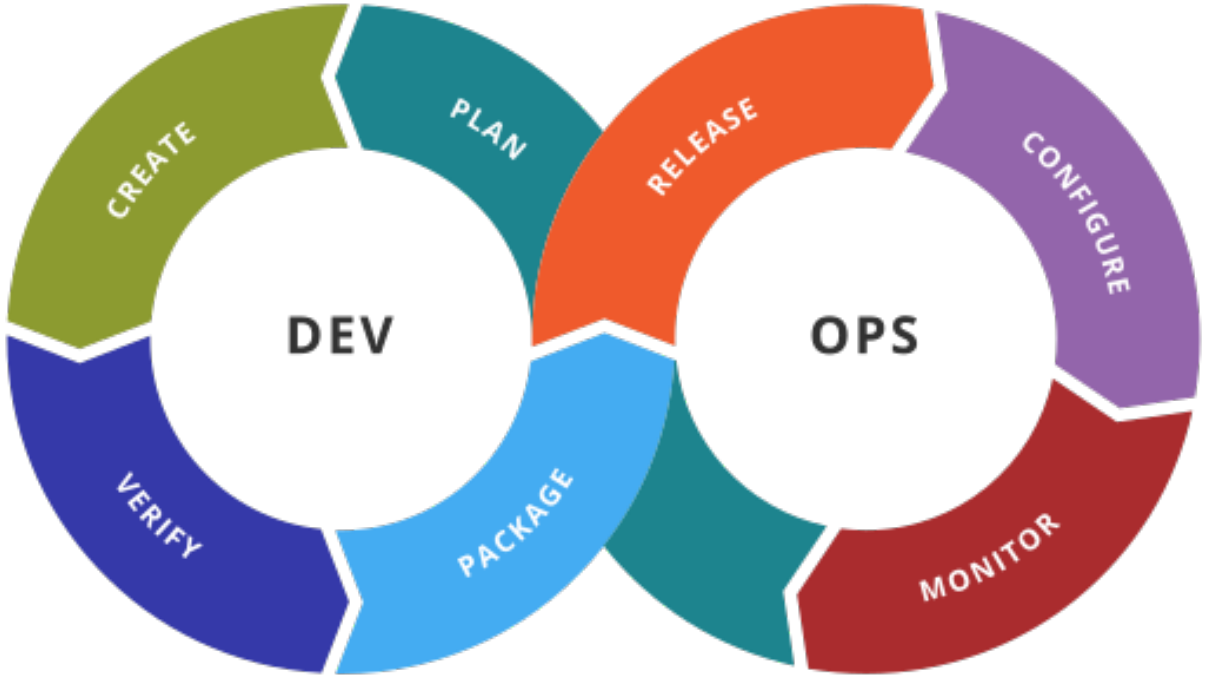
Aren't we missing something?

Agenda

- CI/CD
- **DevOps Overview**
- Continuous Deployment (on the Cloud)
- Infrastructure as Code
- Monitoring

Enter: DevOps

A software engineering practice that aims at **unifying** software development (Dev) and software operation (Ops).



Why DevOps?

- Developers (Dev) and operators (Ops) don‘t always pursue the same goals
  - Devs want to push new features
  - Ops want to keep the system stable and available
- Poor communication between Dev and Ops
- Limited capacity of operations staff
- Limited insight into operations

→ **Leads to slow release schedule**

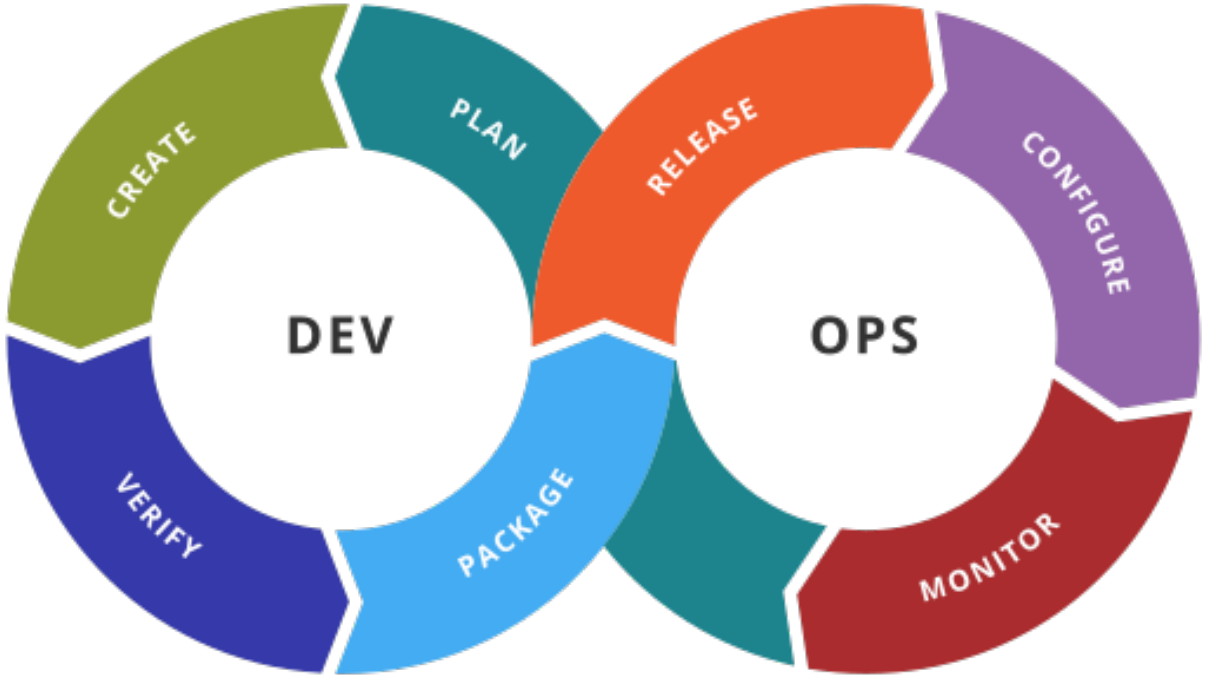
DevOps is about...

- Encouraging ***communication and collaboration between development and operations staff***, get them talking
  - Formally: shared goals and teams of Devs and Ops
  - Informally: beer & chips

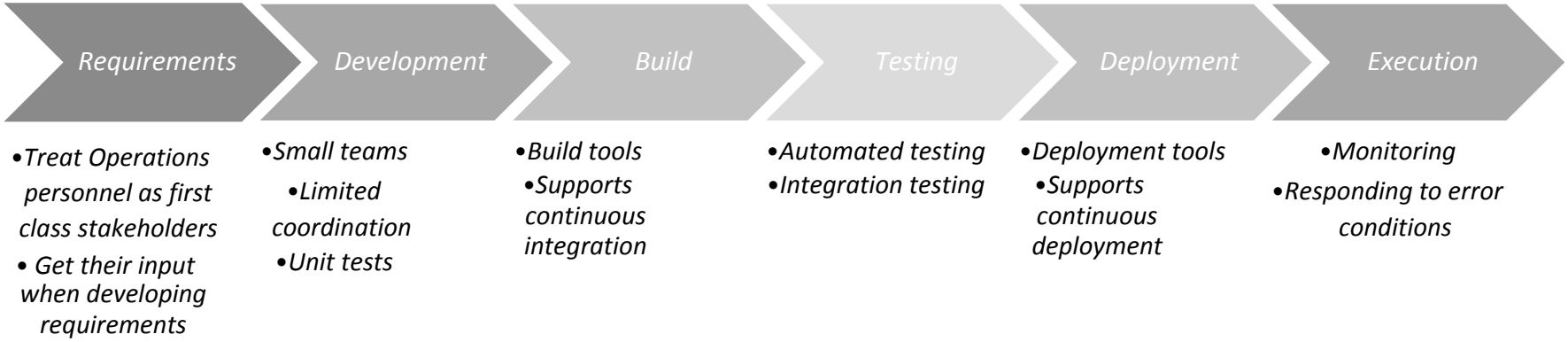
- Bringing “agile” methods to operations

DevOps

A set of practices intended to reduce the **time between committing** a change to a system and the change being **placed into normal production**, while ensuring **high quality**.



DevOps practices (1/2)



- ***Organizational***:
  - *Getting Ops involved in decisions that affect deployment such as logging and monitoring.*
  - *Make Dev more responsible for relevant incident handling*
  - *Enforce strict deployment process and traceability of errors across services*
  - *Have small teams*

DevOps practices (2/2)

- ***Engineering practices:***
  - Use **continuous deployment**, automate everything
    - Commits trigger automatic build, testing, deployment
  - Develop **infrastructure code** with the same set of practices as application code
    - "Infrastructure as Code" : using IaaS APIs, etc., to automate creation of environments
    - Ops scripts are traditionally more ad-hoc
  - **Monitoring** and **Alerting system** for fault identification and reporting

DevOps – main takeaway

- ***Automation and monitoring*** at all steps of software construction…

  …from integration, testing, releasing

  …to deployment and infrastructure management

- ***Developers' responsibility***: unlikely that you'll be able to "throw your final version over the fence" and let **operations** worry about running it!

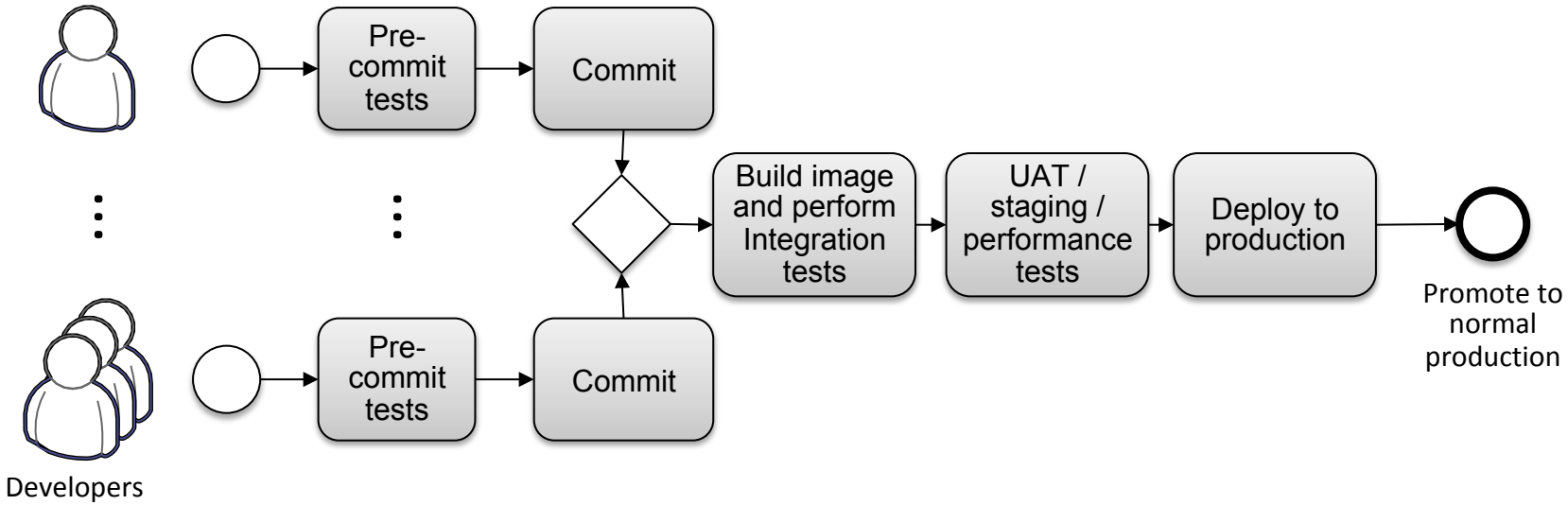
- Result: ***Shorter development cycles***, increased deployment frequency, closer alignment with business objectives

Agenda

- CI/CD
- DevOps Overview
- **Continuous Deployment (on the Cloud)**
- Infrastructure as Code
- Monitoring

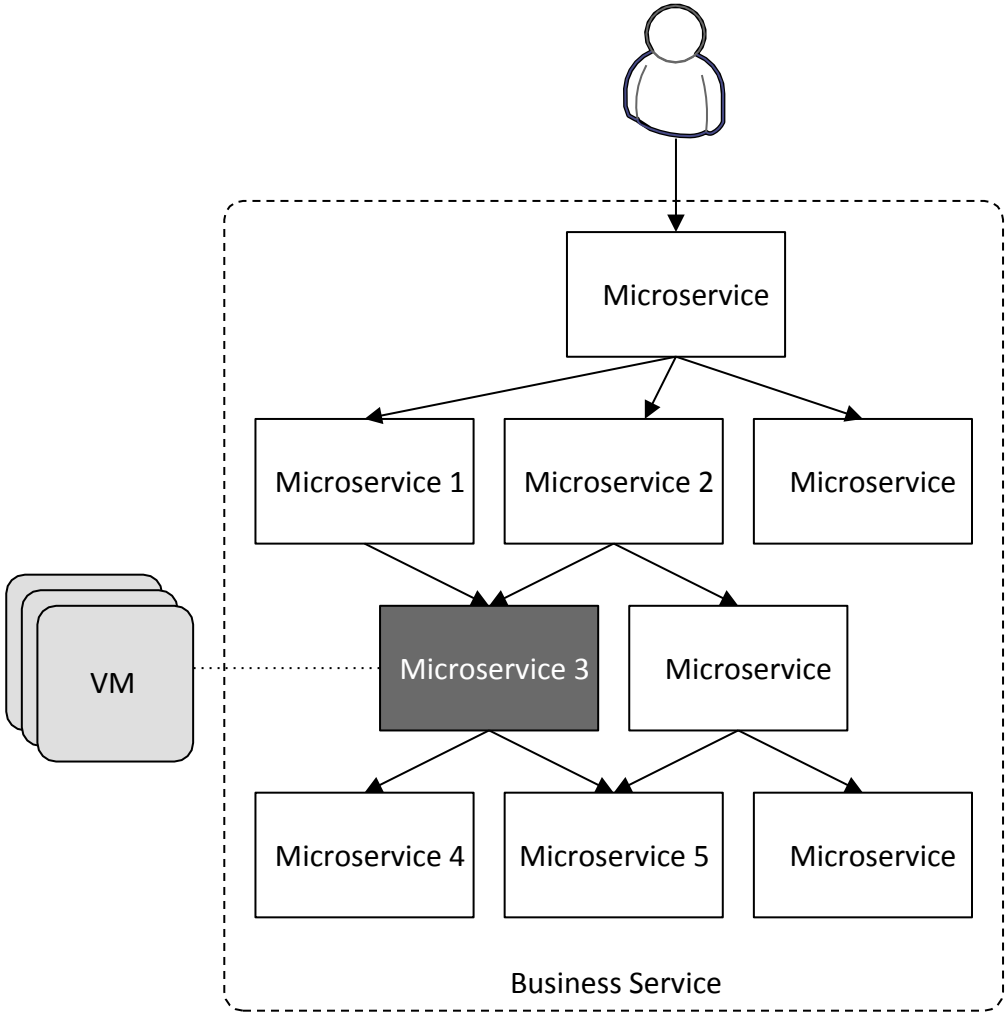
Continuous Deployment

- Deployment pipeline is triggered when intending to commit of code



- All gates from one phase to the next are automatic – else continuous integration / delivery

Deployment is not trivial



Challenges

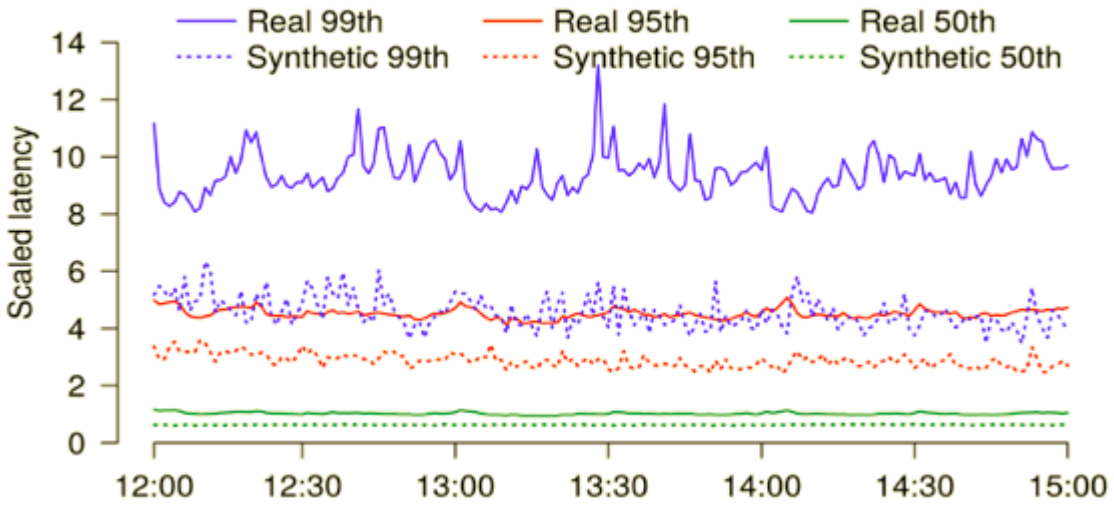
- 24/7 availability is base requirement
- Microservice 3 to be replaced with new version
  - Multiple VMs for it
- Might change how M1 / M2 can use it
- Might change how it uses M4 / M5

Deployment goal and constraints

- Goal: move from current state (N instances of version A of a service 3) to new state (N instances of version B of a service 3)
- Constraints:
  - Any development team can deploy their service at any time – no synchronization among development teams
  - It takes time to replace one instance of version A with an instance of version B (order of minutes)
  - Service to clients must be maintained while the new version is being deployed (24/7 availability)

Testing: why is synthetic testing not enough?

It's very hard to predict user workload

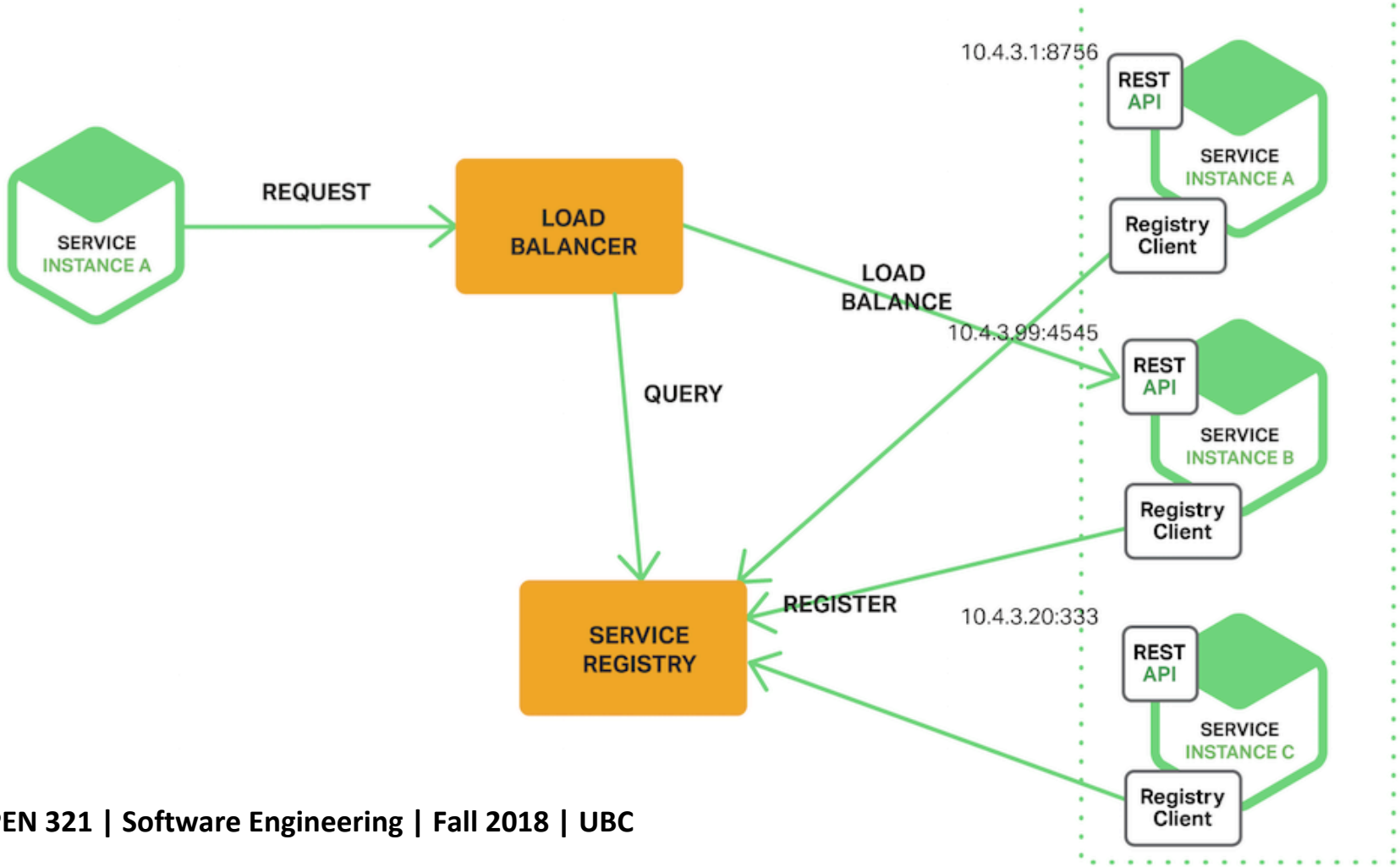


Results of testing Gmail using synthetic vs real data

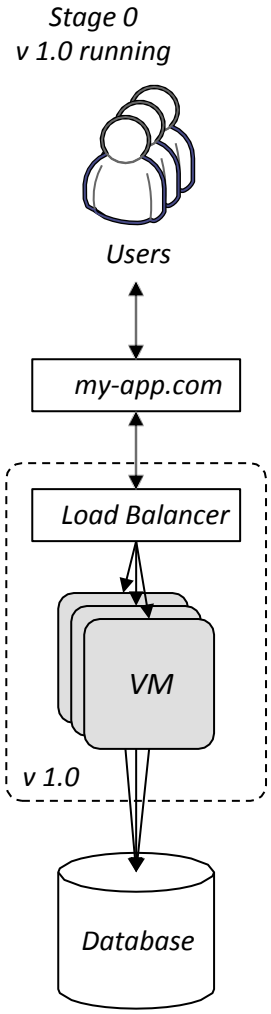
Source: Performance Analysis of Cloud Applications published in NSDI '18

Load Balancer

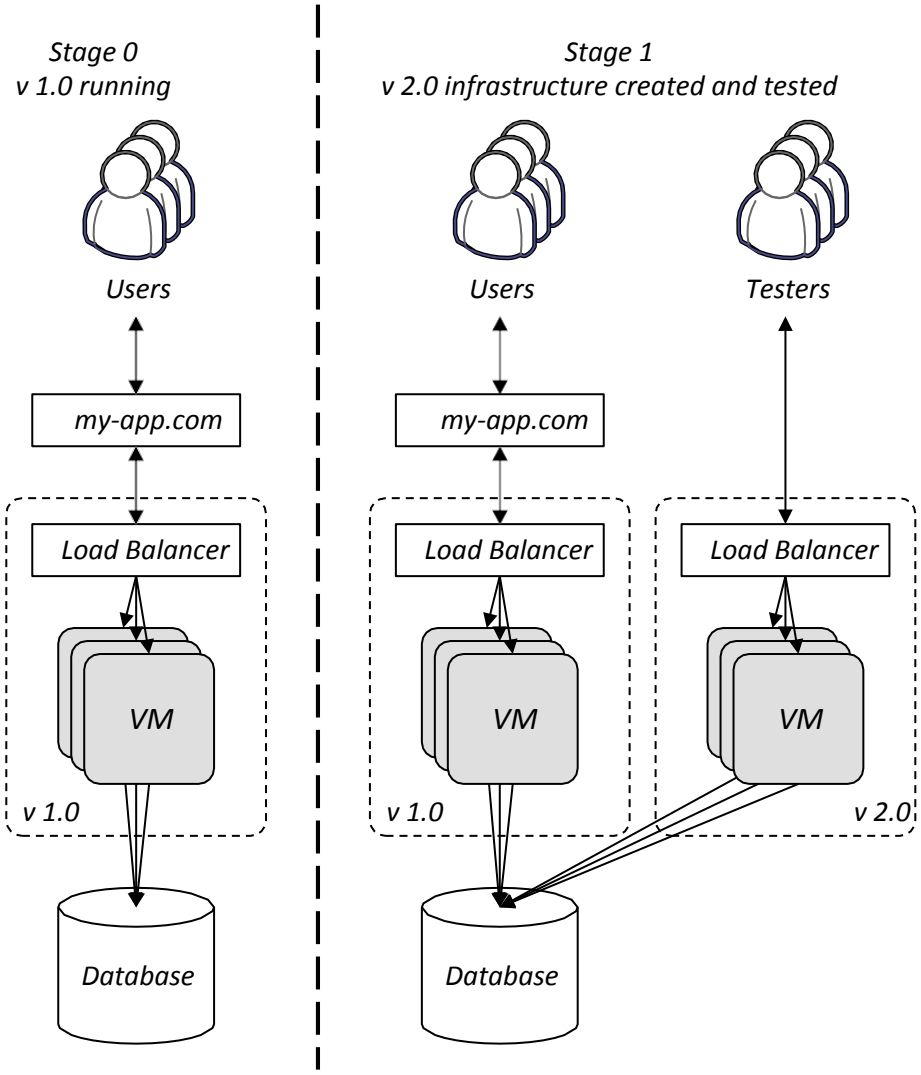
- Routes the request to the appropriate service instance



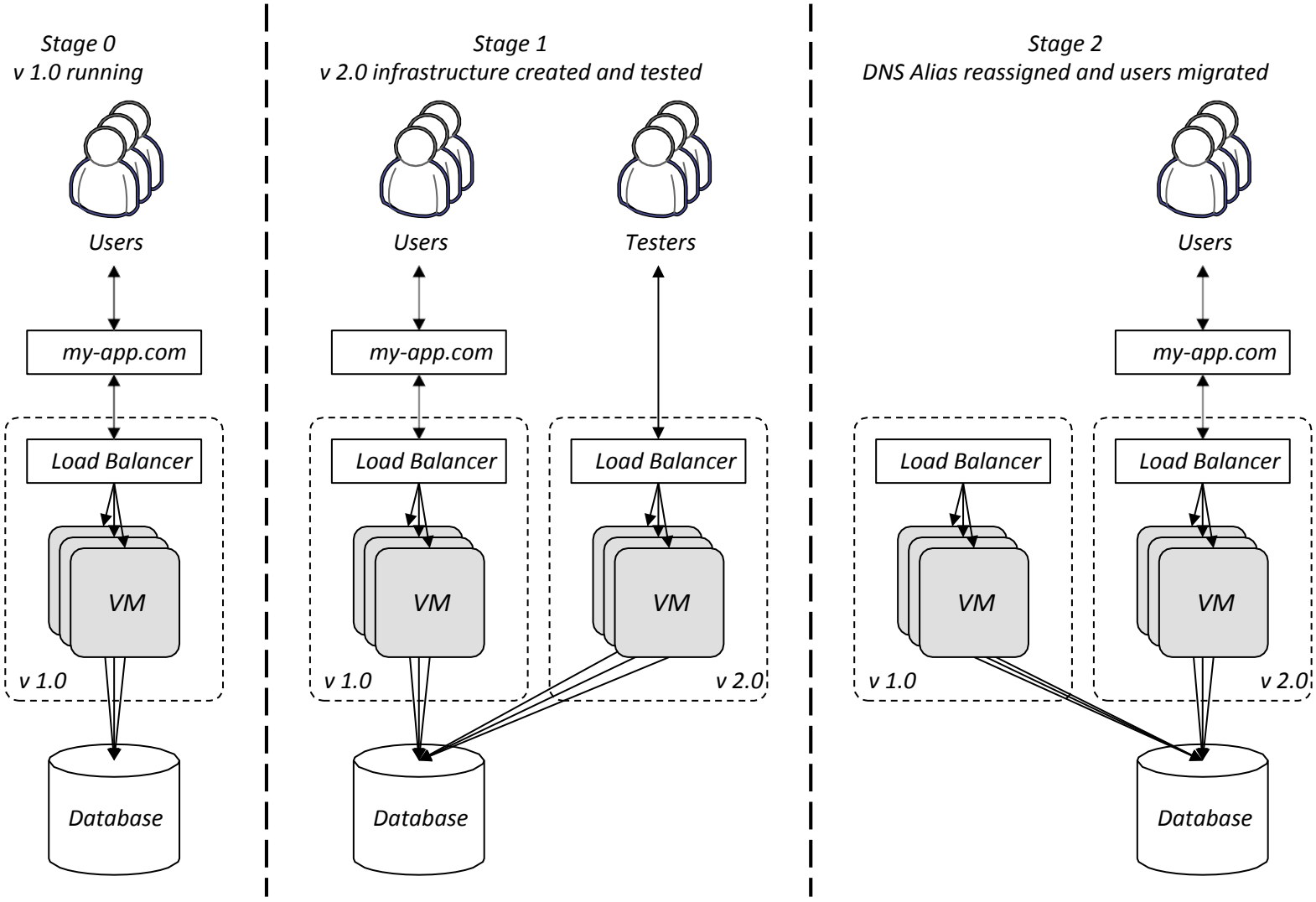
Blue/Green deployment (1/3)



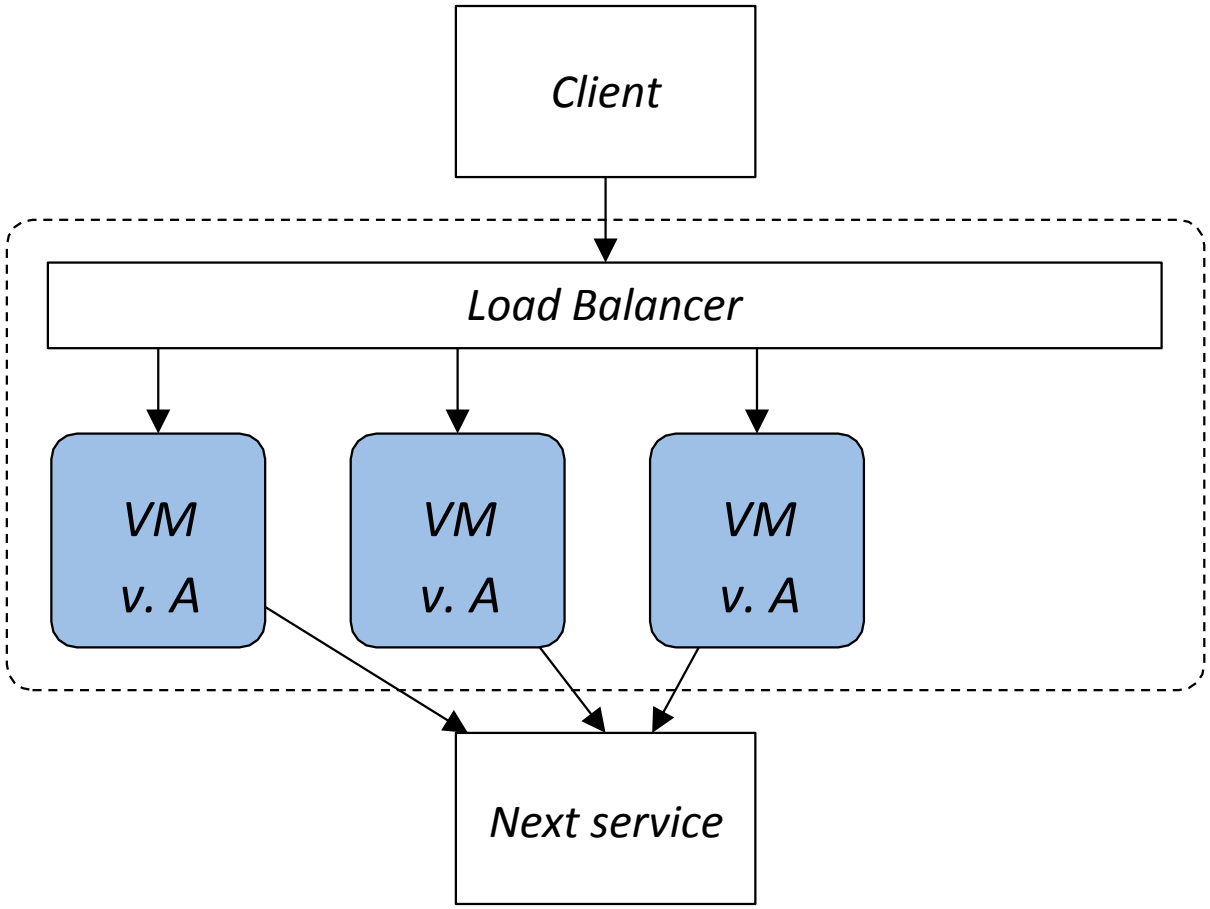
Blue/Green deployment (2/3)



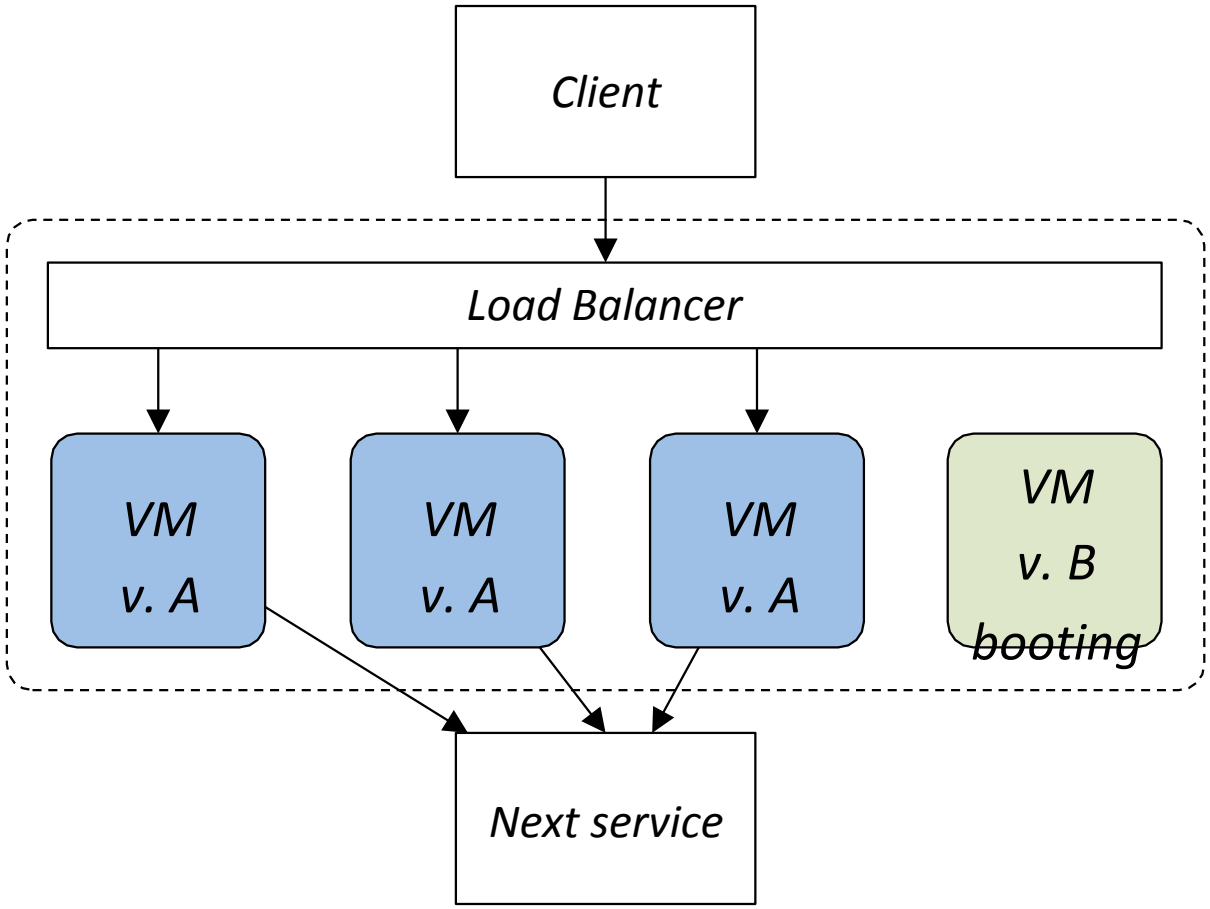
Blue/Green deployment (3/3)



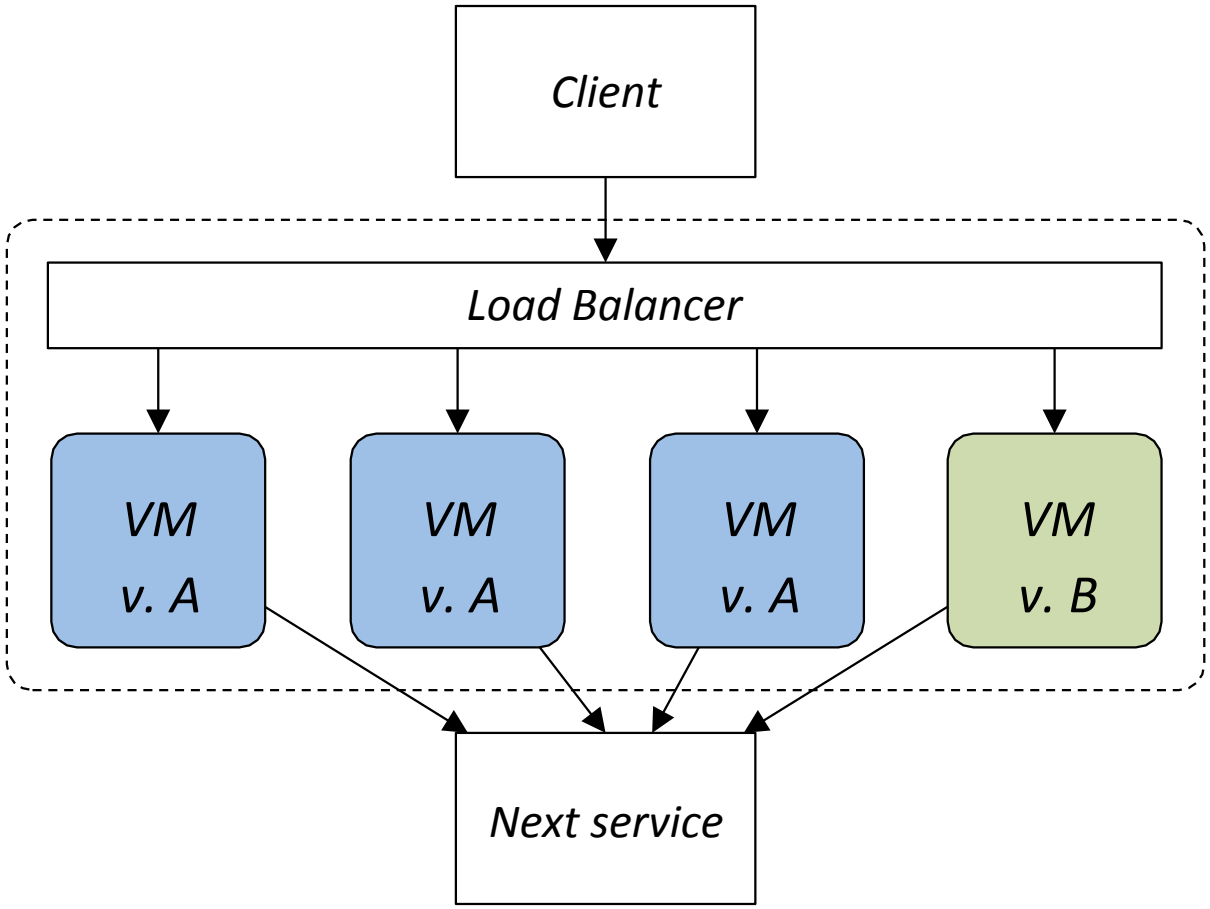
Rolling Upgrade



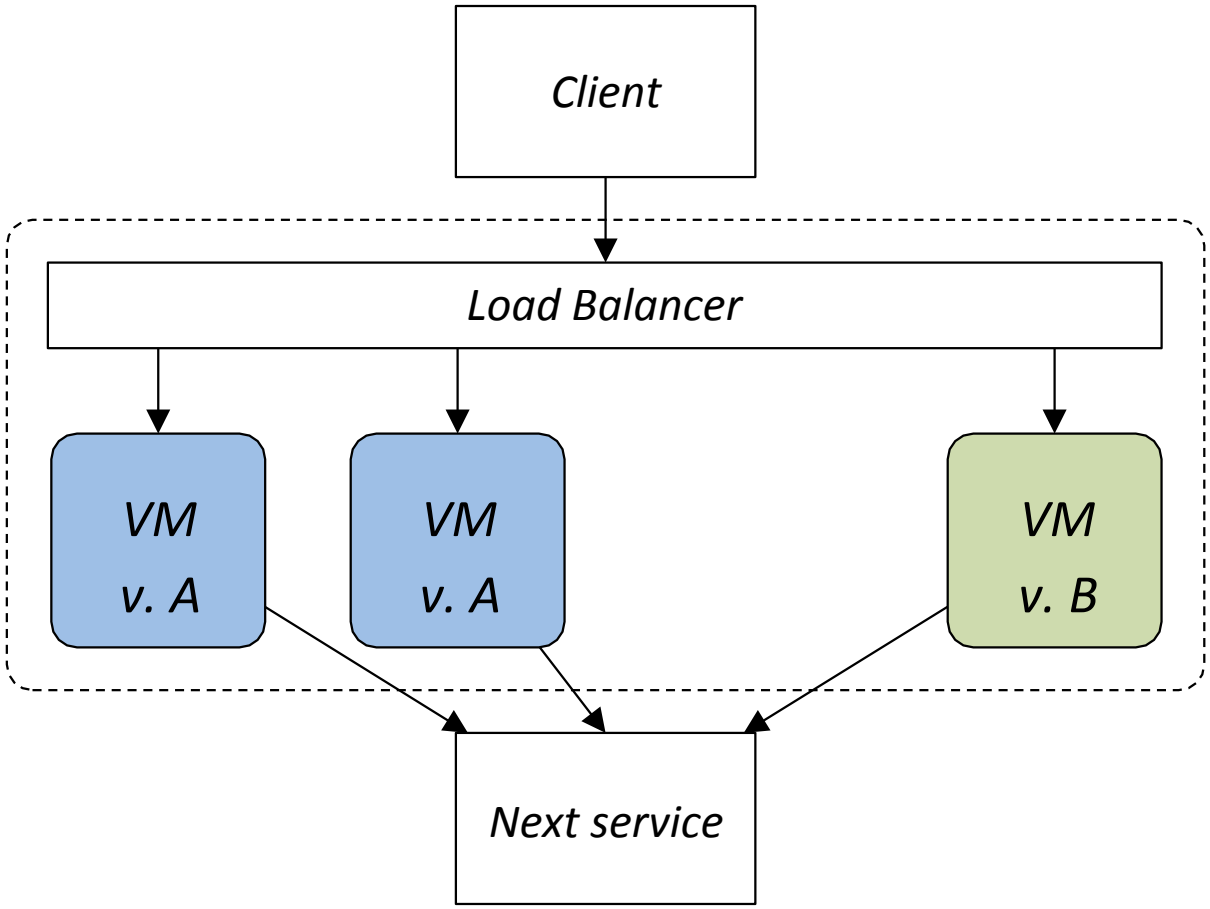
Rolling Upgrade



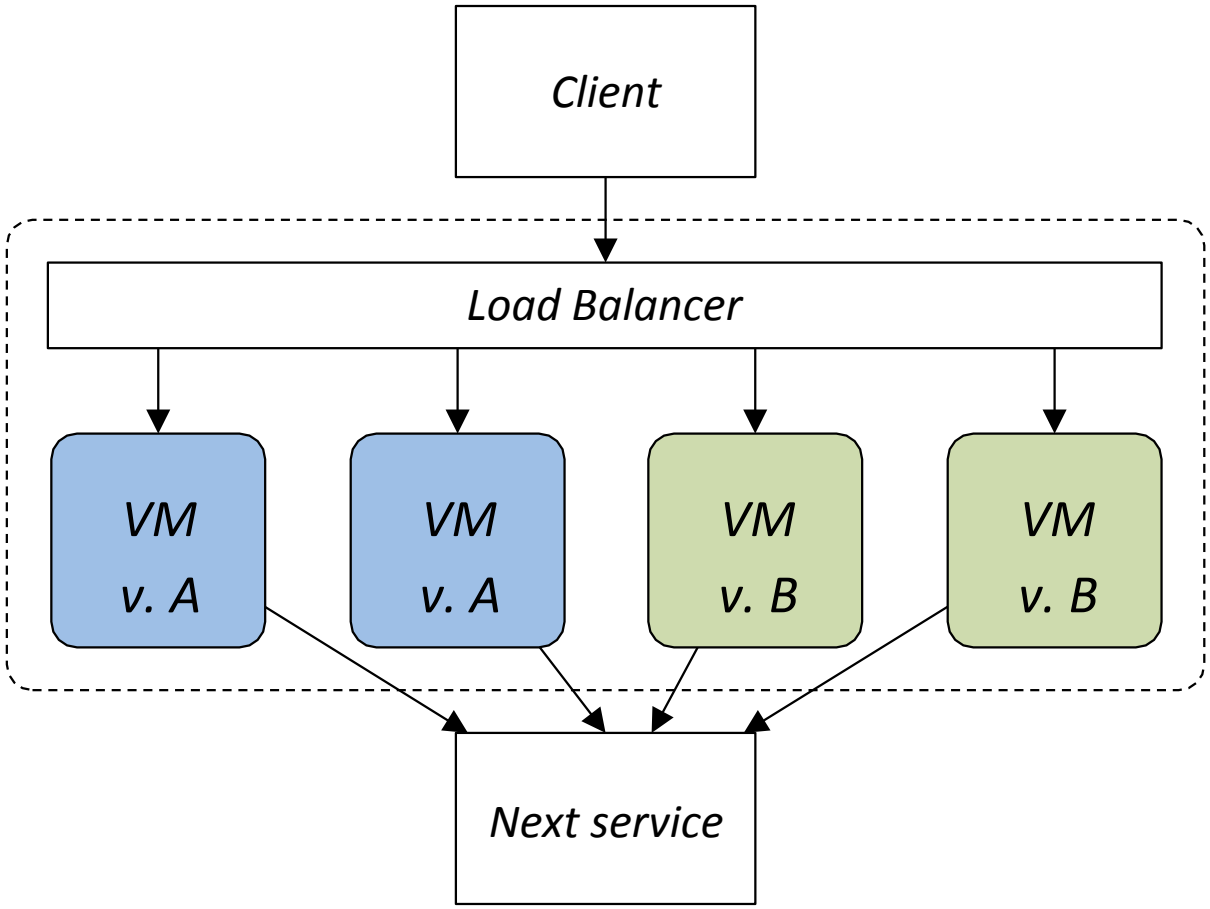
Rolling Upgrade



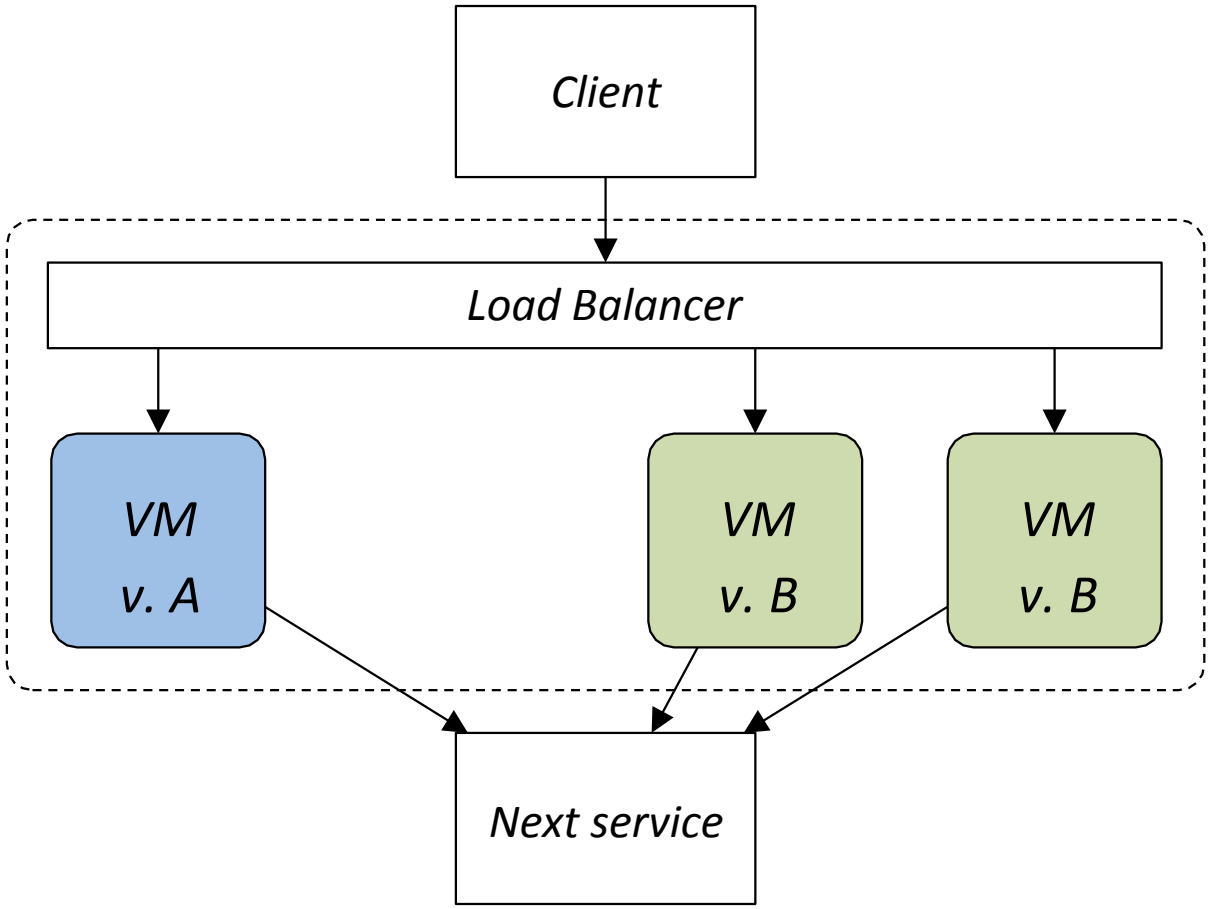
Rolling Upgrade



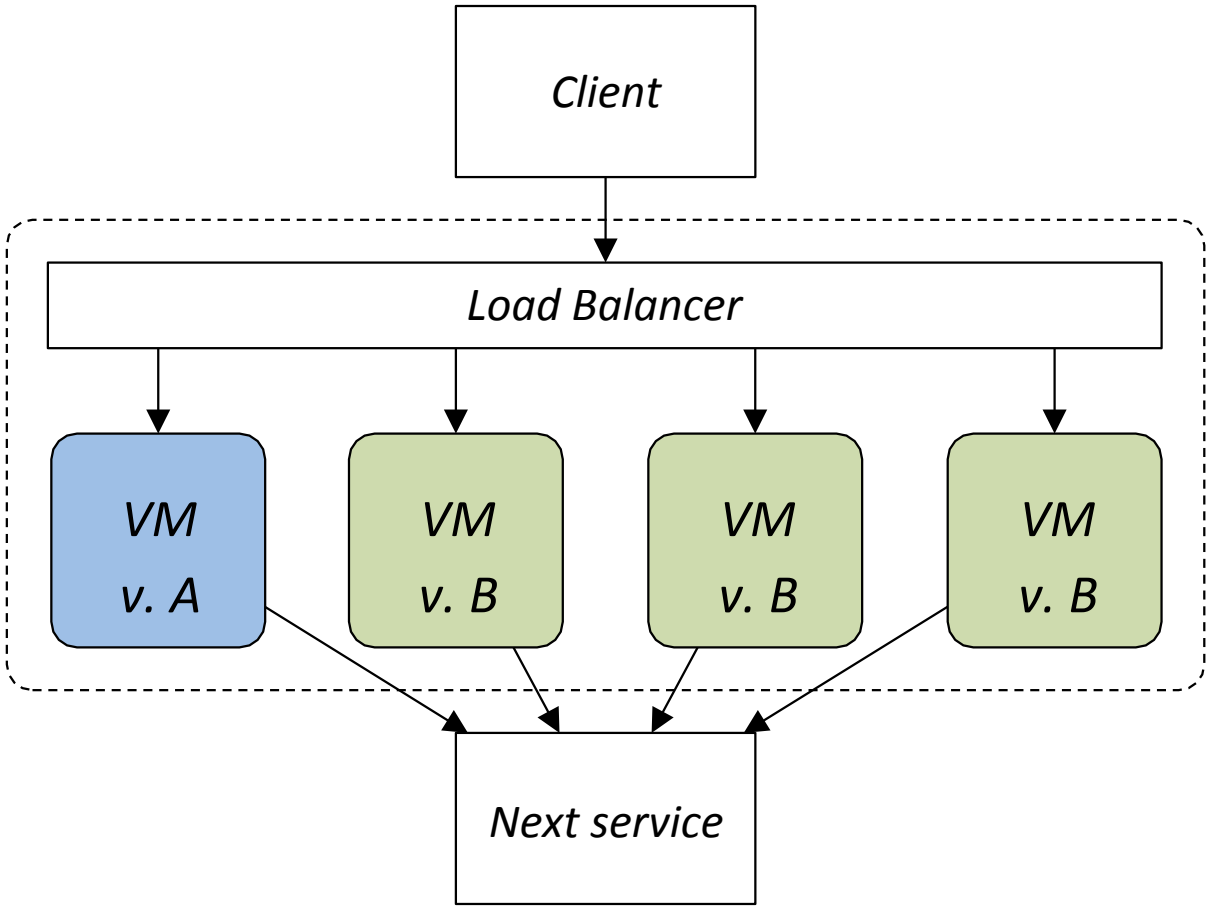
Rolling Upgrade



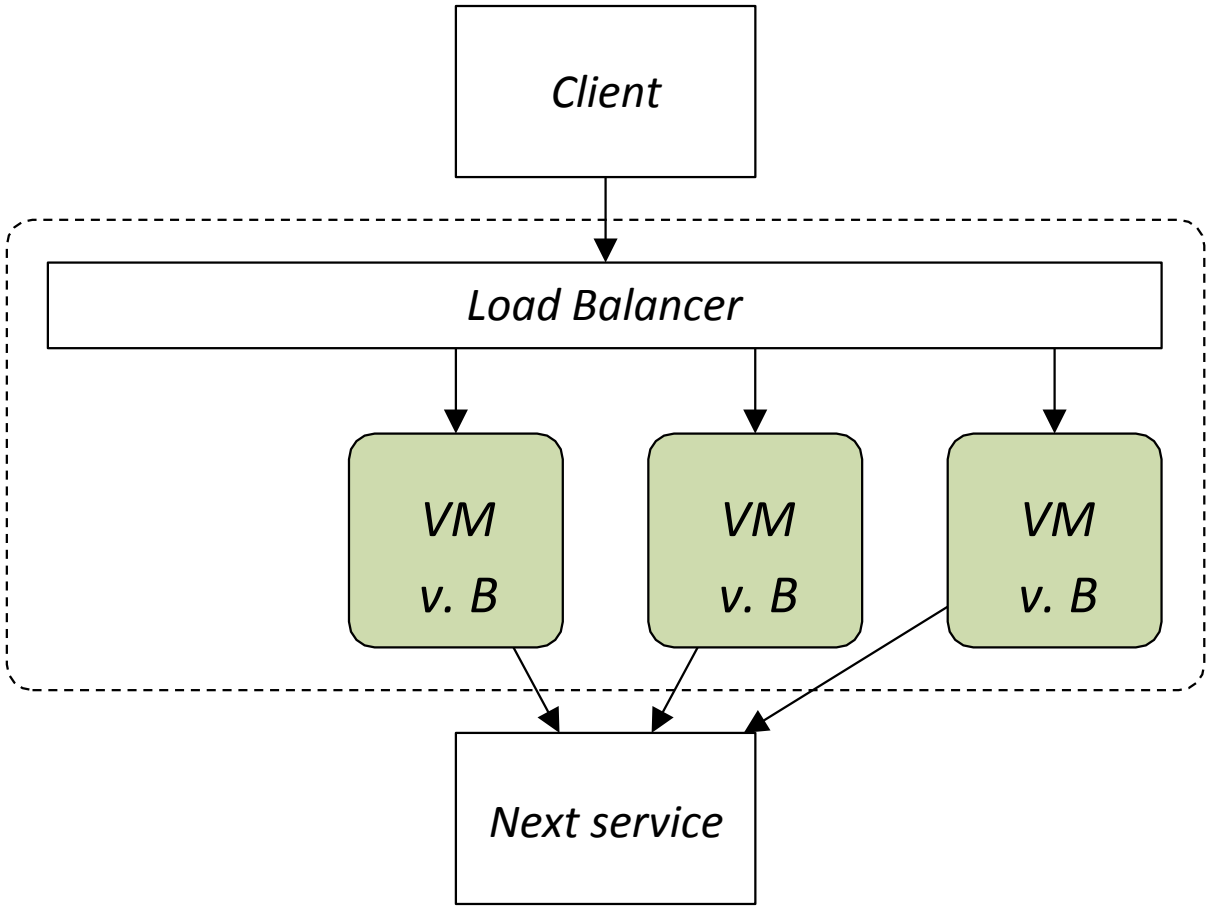
Rolling Upgrade



Rolling Upgrade



Rolling Upgrade



Blue/Green vs. Rolling Upgrade

Blue/Green Deployment

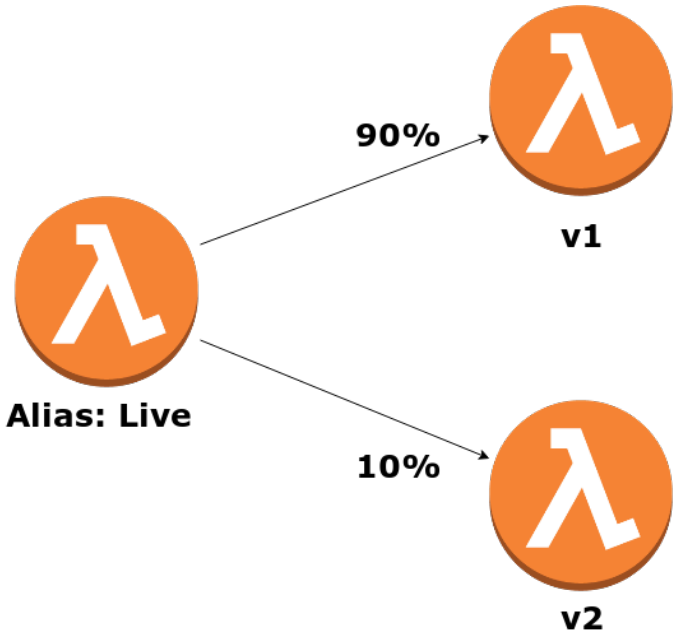
- Only one version available to the client at any time
- Requires 2N VMs
  - additional cost
- Rollback is easy

Rolling Upgrade

- Multiple versions are available for service at the same time
- Requires N+1 VMs
  - Can be done at nearly no extra cost

Canary testing

- Canaries are a small number of instances of a new version placed in production in order to perform live testing in a production environment.
- Canaries are observed closely to determine whether the new version introduces any logical or performance problems. If not, roll out new version globally. If so, roll back canaries.



Implementation of canaries

- Create set of new VMs as canaries (they are unaware of that)
- Designate a collection of customers as testing the canaries. Can be, for example
  - organization-based
  - geographically based
  - at random
- Then
  - Route messages from canary customers to canaries by making registry/load balancer canary aware
  - Observe the canaries closely and decide on rolling out / back

Agenda

- CI/CD
- DevOps Overview
- Continuous Deployment (on the Cloud)
- **Infrastructure as code**
- Monitoring

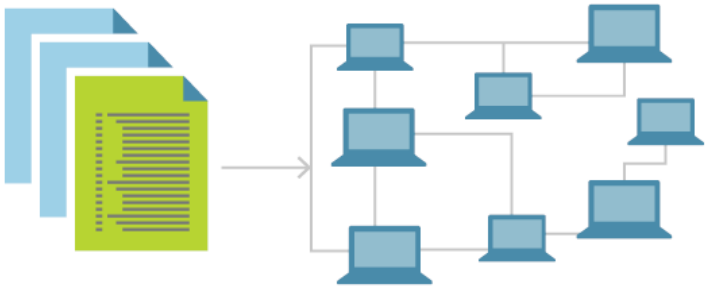
Infrastructure management

Amazon AWS Infrastructure as Code 2017

What is "Infrastructure as Code"?

- Infrastructure as Code (IaC) is the management of infrastructure using the same tools used by DevOps.

- Use definition files instead of traditional interactive configuration tools.



- Idempotence: IaC should generate the same environment every time it is applied.

IaC

Advantages:

- Single source of truth
- Increase repeatability and testability
- Decrease provisioning time
- Rely less on availability of persons to perform tasks

IaC: Examples

```
"apiVersion": "2017-03-30",
"type": "Microsoft.Compute/virtualMachines",
"name": "[variables('vmName')]",
"location": "[parameters('location')]",
"dependsOn": [
    "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
    ],
"properties": {
    "hardwareProfile": {
      "vmSize": "[parameters('virtualMachineSize')]"
    },
    "osProfile": {
      "computerName": "[variables('vmName')]",
      "adminUsername": "[parameters('adminUsername')]",
      "adminPassword": "[parameters('adminPassword')]"
    },
    "storageProfile": {
      "imageReference": {
        "publisher": "[variables('imagePublisher')]",
        "offer": "[variables('imageOffer')]",
        "sku": "[variables('ubuntuOSVersion')]",
        "version": "latest"
      },
      "osDisk": {
        "caching": "ReadWrite",
        "createOption": "FromImage",
        "diskSizeGB": "100"
      }
    },
    "networkProfile": {
      "networkInterfaces": [
        {
          "id": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]"
        }
      ]
    }
}
```

Servers: cattle vs. pets

*Pets vs Cattle

Is a widely used metaphor for how IT operations should handle servers in the cloud.

Servers are like pets

You name them and when they get sick, you nurse them back to health

Servers are like cattle

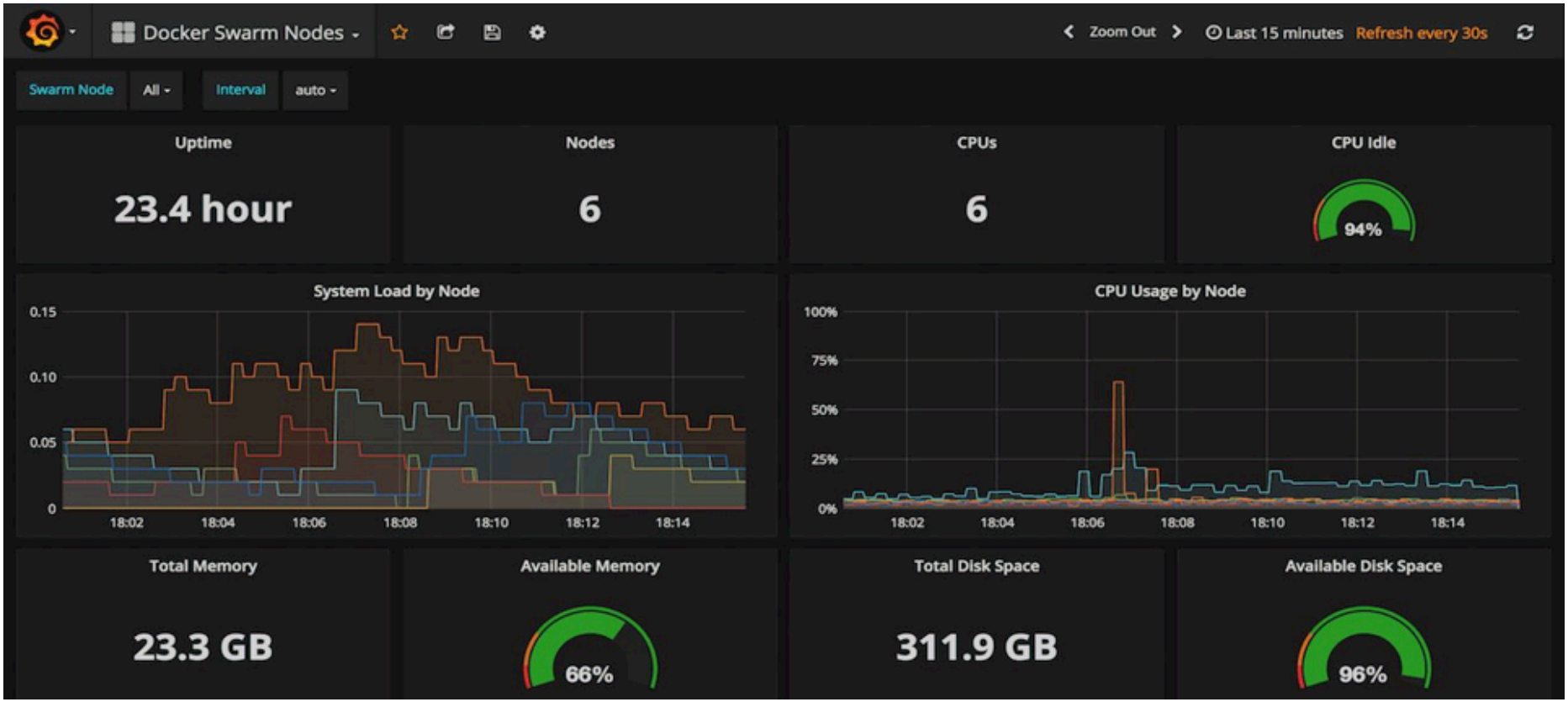
You number them and when they get sick, you get another one

https://blogs.msdn.microsoft.com/mvpawardprogram/2018/02/13/infrastructure-as-code/

Agenda

- CI/CD
- DevOps Overview
- Continuous Deployment (on the Cloud)
- Infrastructure as Code
- **Monitoring**

Monitoring


Docker Swarm Nodes
Zoom Out
Last 15 minutes
Refresh every 30s
Swarm Node
All
Interval
auto
Uptime
23.4 hour
Nodes
6
CPUs
6
CPU Idle
94%
System Load by Node
0.15
0.10
0.05
0
18:02
18:04
18:06
18:08
18:10
18:12
18:14
CPU Usage by Node
100%
75%
50%
25%
0%
18:02
18:04
18:06
18:08
18:10
18:12
18:14
Total Memory
23.3 GB
Available Memory
66%
Total Disk Space
311.9 GB
Available Disk Space
96%

What data you can monitor

- System Resource usage metrics
  - CPU – Utilization, Load, CPI
  - Memory – Usage, Bandwidth, page miss rate
  - Disk – Usage, Bandwidth
  - Network – Rx Bandwidth, Tx Bandwidth
  - Cache – Miss rate
- User performance metrics
  - Response time
  - Throughput (QPS)
  - Error rate – 5XX, 4XX

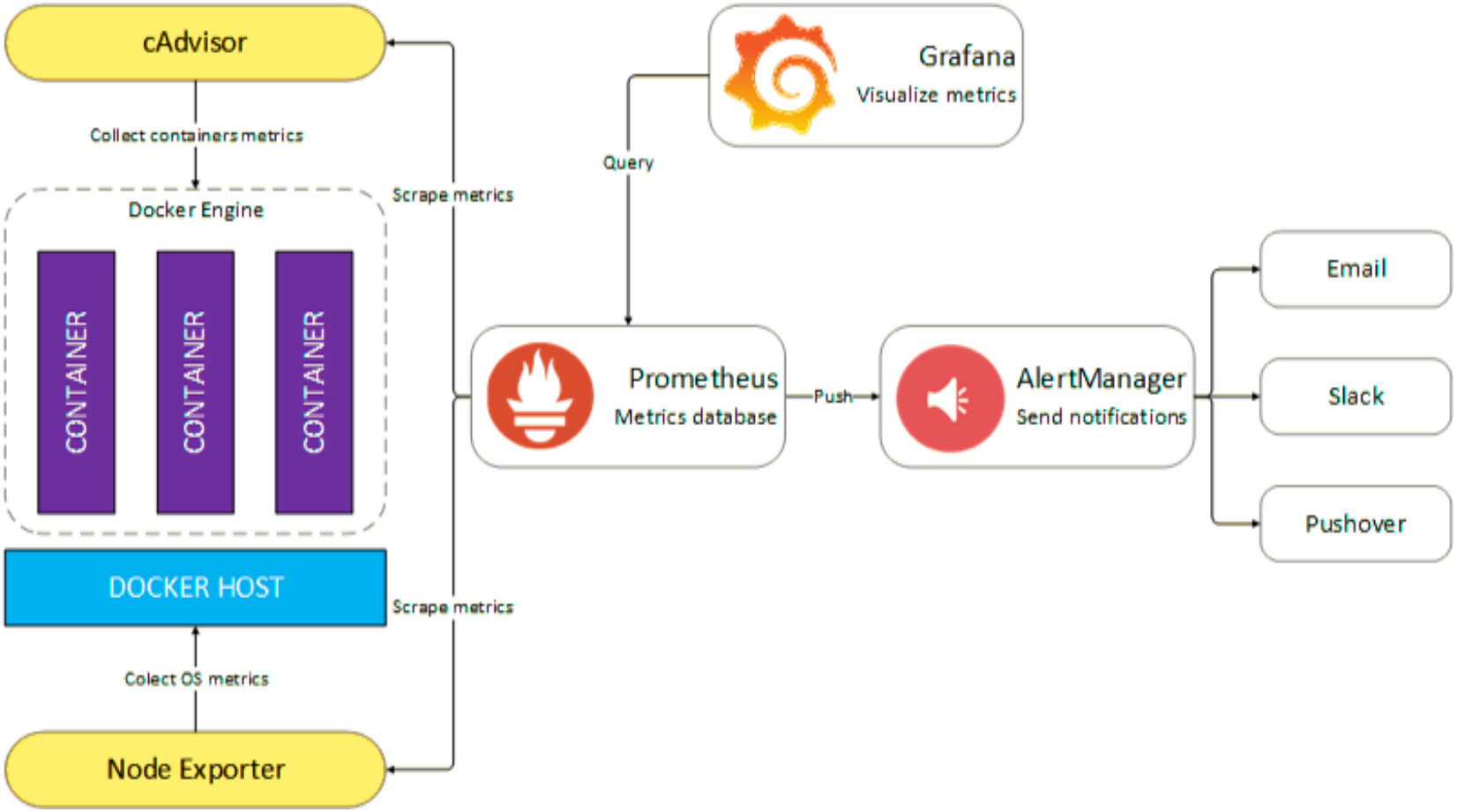
Alerts

You want to get notified (through mail or sms) when unusual events happen in your system.

How to monitor microservices

1. **Metrics exporters**
   a. **Cadvisor** - Gets container metrics such as cpu usage etc.
   b. **Node-exporter** - Gets server metrics from each machine.
   c. **Instrumentation/Telemetry** code for custom telemetry - Gets application specific metrics such as response time of APIs
2. **Metrics database**
   a. **Time series databases** - Metrics are ordered by timestamp - e.g. Prometheus
3. **Metric consumers**
   a. **Visualization** - Creating dashboards e.g. Grafana
   b. **Alerts** - Report unusual events.

How to monitor microservices



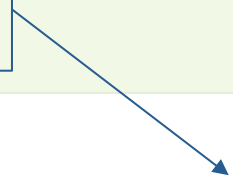
Instrumentation and telemetry

You have to instrument the application code

```
start := time.Now()

// Do something

duration := time.Since(start)
code := 200 // some HTTP Code
histogram.WithLabelValues(fmt.Sprintf("%d",
code)).Observe(duration.Seconds())
```



Send update to metrics database

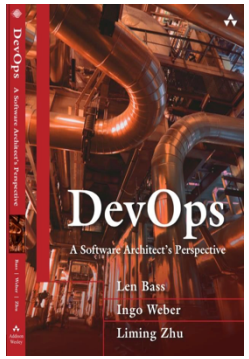
Summary

- DevOps
  - Developers and operators collaborating to joint goal
  - Fast deployment of new features to production while ensuring high quality
- Continuous Deployment
  - Deployment without downtime, e.g. Blue/Green, Rolling Upgrade
  - Live testing

More Info

- https://devops.com/

- “DevOps: A Software Architect's Perspective”, L. Bass, I. Weber, L. Zhu, Addison-Wesley Professional, 2015.



- AWS What is Devops?