DevOps on AWS

Building Systems to Deliver Faster


TELEPHONE
87
Software moves
faster today

15yrs

The **average lifespan of an S&P company** dropped from 67 years in the 1920s to 15 years today

2/3

More than two-thirds of **IT budgets** go toward **keeping the lights on**

77%

of **CEOs believe security risk has increased** in the last few years and **65%** believe their **risk management** capability is **falling behind**

[SOURCES]
15yrs- http://www.theatlantic.com/business/archive/2015/04/where-do-firms-go-when-they-die/390249/
2/3 of IT budget- The Midyear Global Tech Market Outlook For 2015 To 2016, Forrester Research
66% of CEOs- 2015 CEO Survey: Committing to Digital, Gartner

How This Affects You

You're left **without the necessary resources** to pursue critical business initiatives required to maintain a competitive advantage

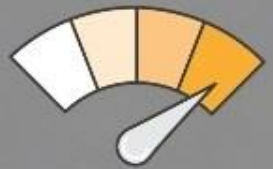
Your traditional IT model **lacks the agility** you need to keep pace with market disruptors

Insufficient security, compliance and **availability** can hamper your ability to compete and open the door to sophisticated, hard-to-identify attacks

aws

Responding requires a new model

Focus on differentiating your company



Innovate at speed

Reduce risk

aws

Why does DevOps matter?

5x

Lower change failure rate

440x

Faster from commit to deploy

46x

More frequent deployments

44%

More time spent on new features and code

Source: Puppet 2017 State of DevOps Report

aws

“Finding time for innovation is hard,

we’re just ***too busy****…”*

aws

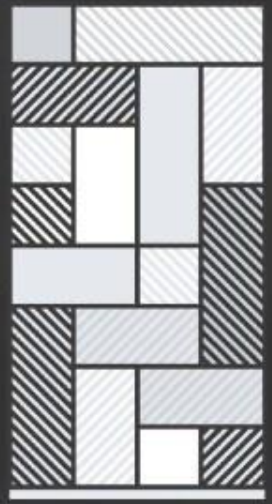
*“But, we have **too much legacy**…”*

https://secure.flickr.com/photos/pixelthing/15806918992/


Amazon.com used to be a monolith...
A 1GB executable that took 18hrs to compile, with a centralised deployment team

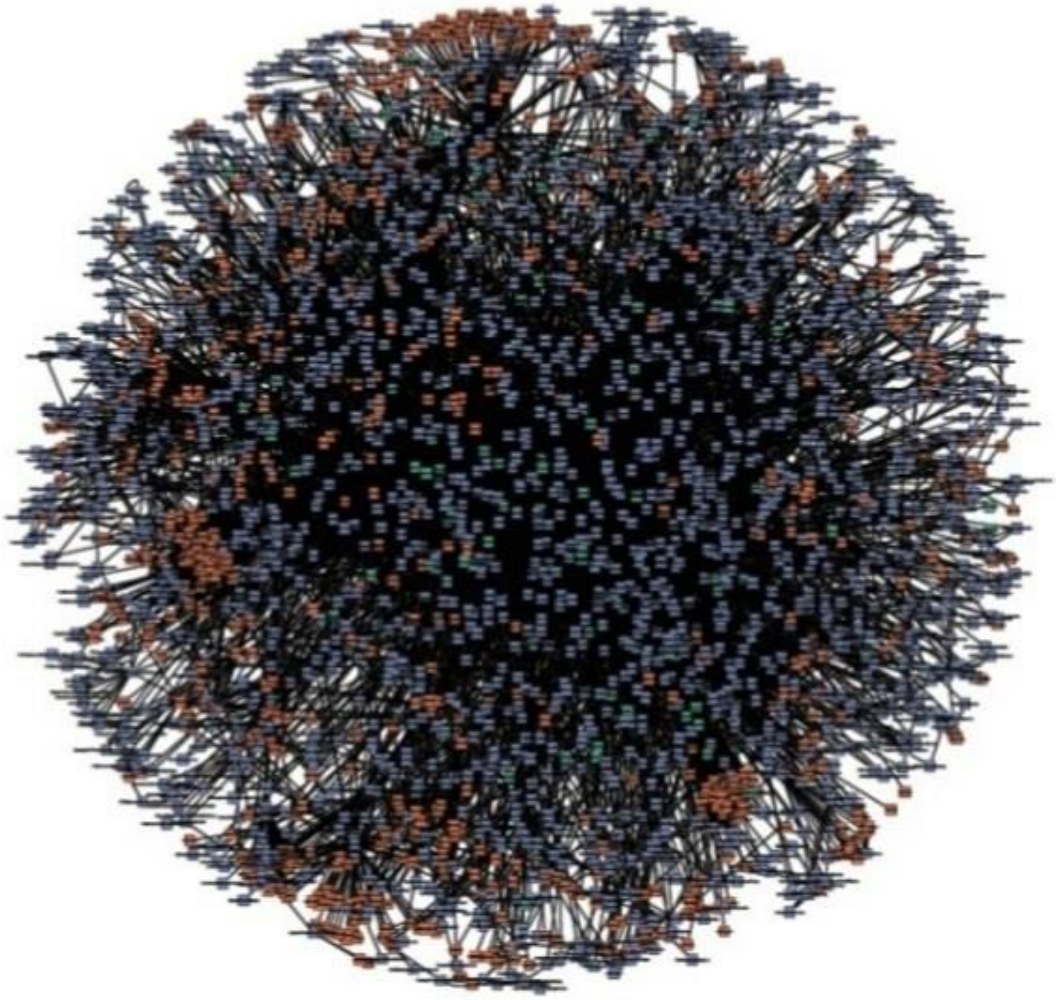
Development transformation at Amazon: 2001-2009

2001



2006

2009

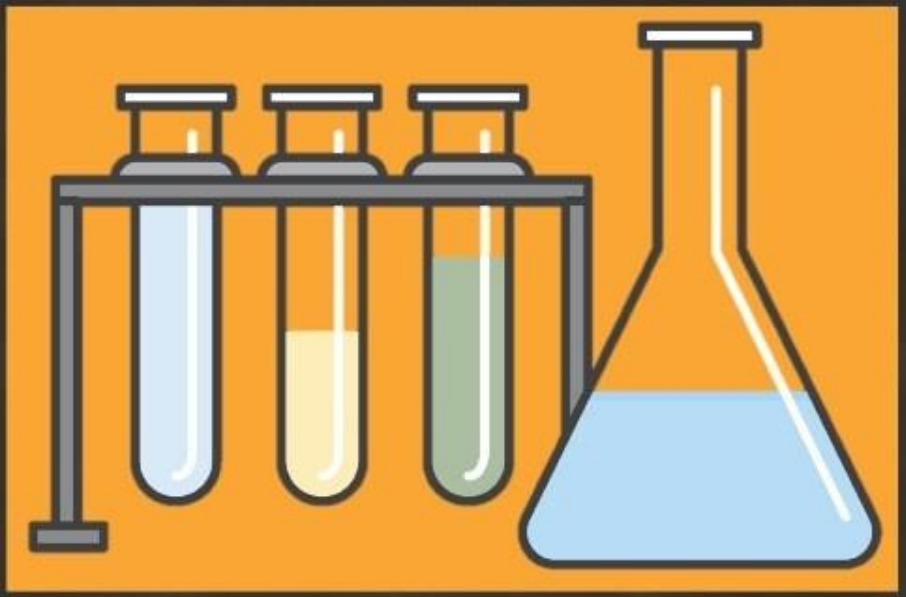


- Single-purpose
- Connect only through APIs
- Connect over HTTPS
- Largely “black boxes” to each other
- “Microservices”

aws

Things went much better under this model and teams were developing features faster than ever, but we felt that we could still improve.

aws

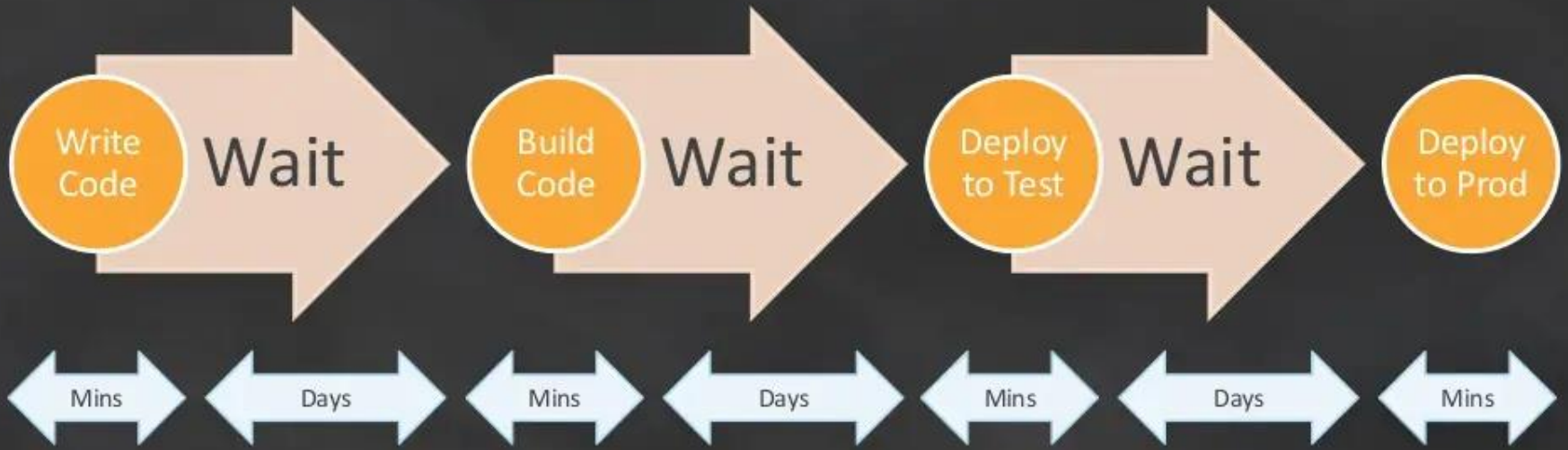


In 2009, we ran a study to find out where inefficiencies might still exist. We found that many teams were still being slowed down by manual processes and work flows.

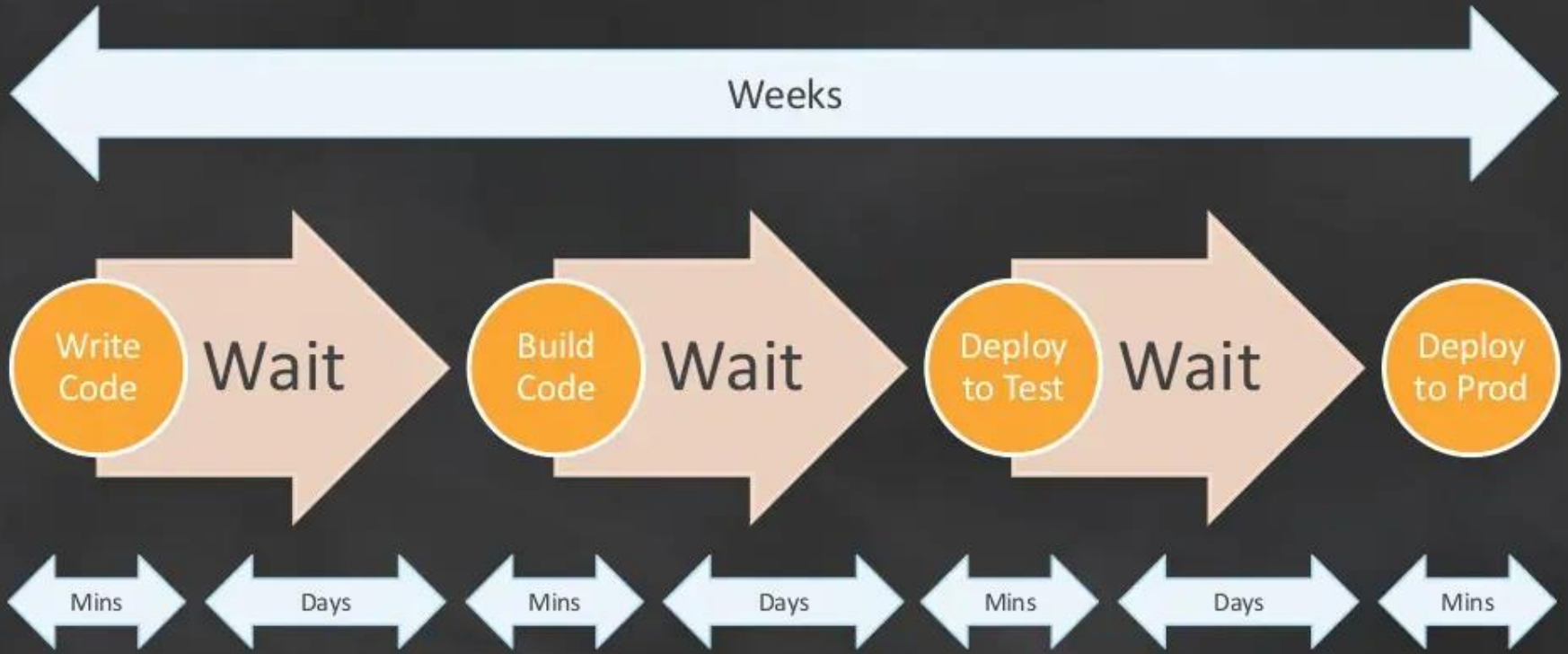
aws

We were just waiting.

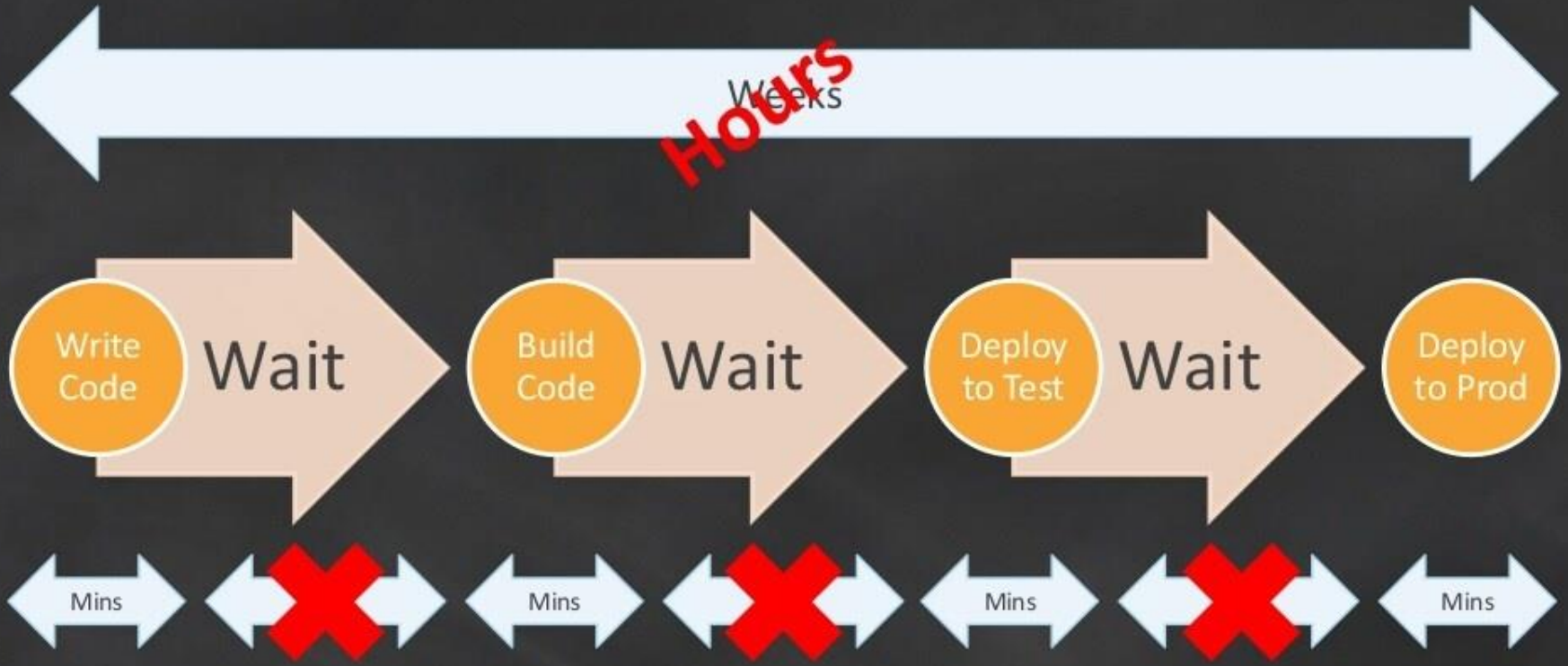
We were just waiting.

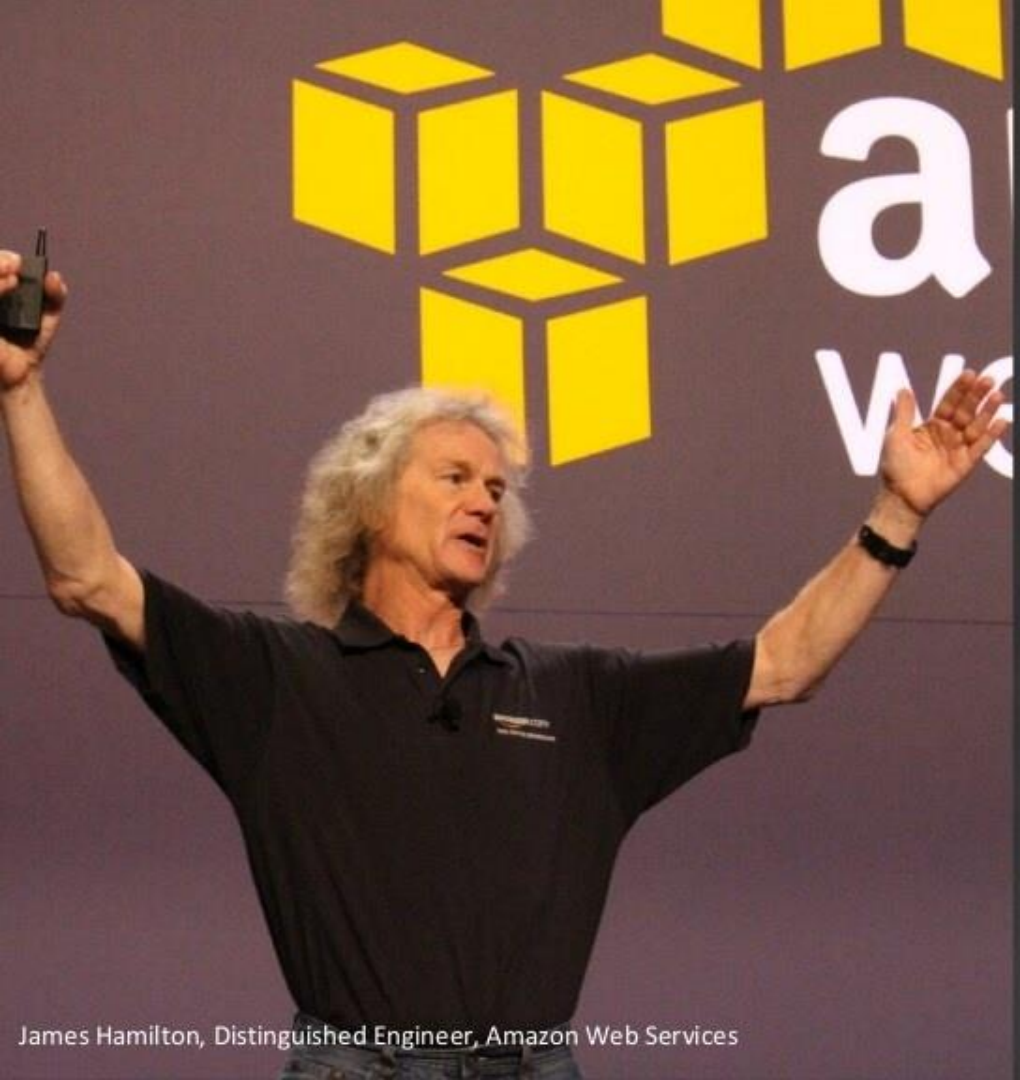


We were just waiting.



We were just waiting.





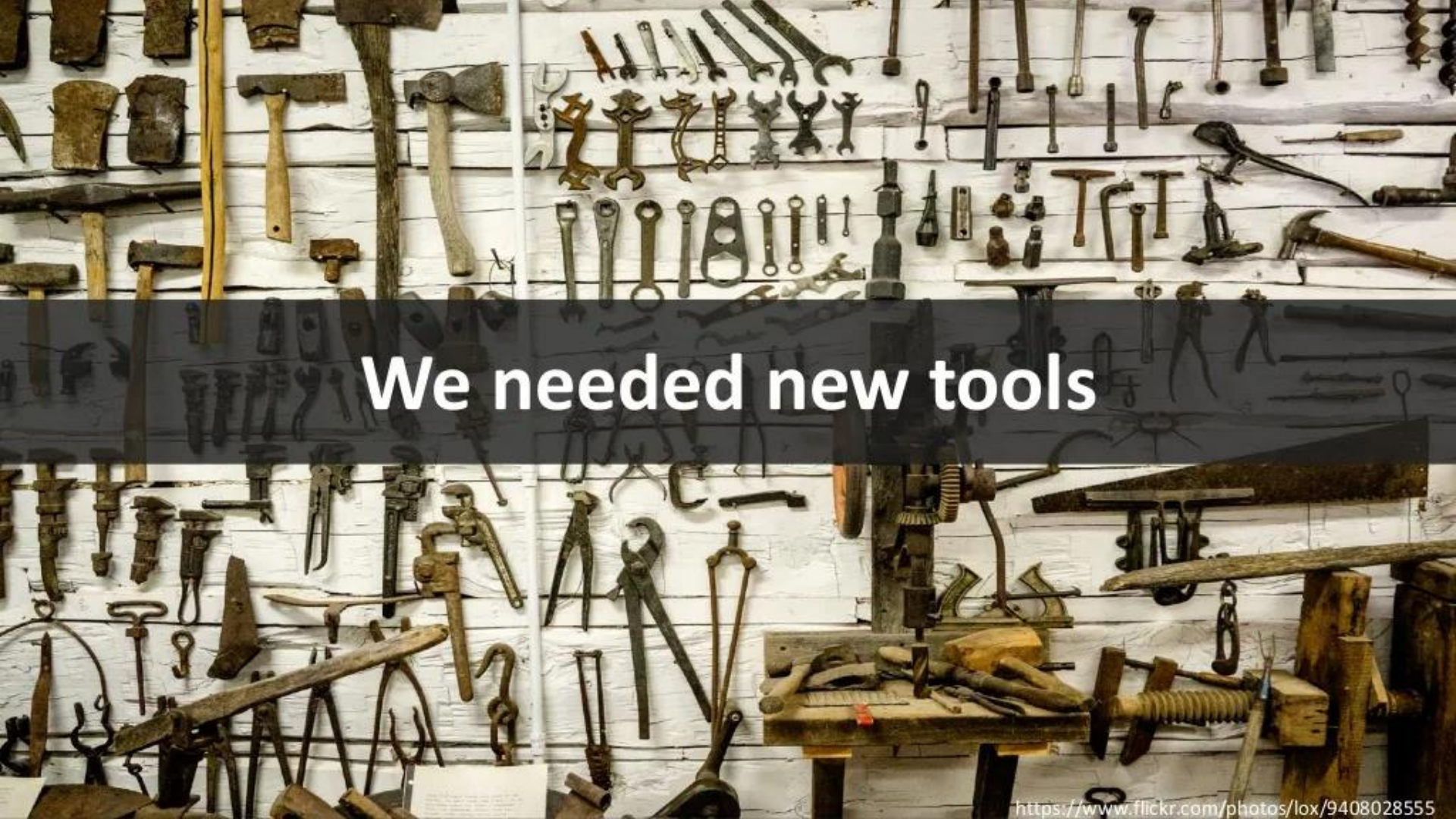
James Hamilton, Distinguished Engineer, Amazon Web Services

"We have long believed that 80% of operations issues originate in design and development…

Most operations issues, however, either have their genesis in design and development or are best solved there.

If the development team is frequently called in the middle of the night, automation is the likely outcome. If operations is frequently called, the usual reaction is to grow the operations team."

aws


We needed new tools
https://www.flickr.com/photos/lox/9408028555

- Self-service
- Technology-agnostic
- Encourage best practices
- Single-purpose services

aws

- Deployment service
- No downtime deployments
- Health checking
- Versioned artifacts and rollbacks

aws

Automated actions and transitions; from check-in to production

Development benefits:

- Faster
- Safer
- Consistent & Standardized
- Visualization of the process

aws

Microservice development lifecycle

General Practices followed by Amazon Developers

- CI/CD is a MUST!
  - Commit frequently
  - Builds on every commit
  - Build once in a given execution flow
  - Deploy to a running environment for further testing

- Everything as a code (application, infrastructure, documentation) goes into a repository
  - If its not in a repository, it doesn't go into production environments!
- Start with continuous delivery ("gated" promotion) and build up to continuous deployment once evidence of a high-level of excellence in testing is clear
- Deploy to canaries, test, deploy to an AZ, test, deploy to a Region, test

aws

Thousands of teams

× Microservice architecture

× Continuous delivery

× Multiple environments

= 60 million deployments a year

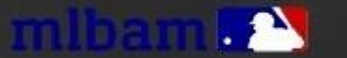
= 1.9 deployments / second

What is DevOps anyways?

Cultural Philosophy + Practices + Tools

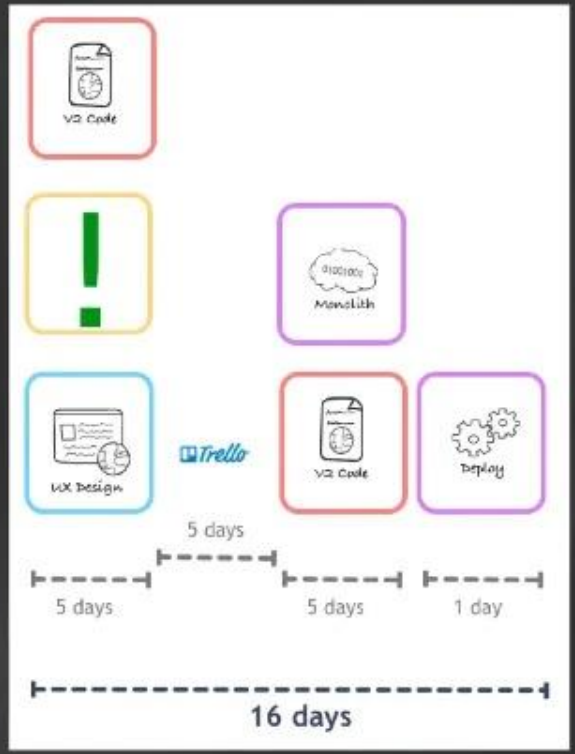
DevOps – Don't Take It Just From Us:

- **Oscar Health**: 2 systems engineers, 45+ Developers, self service infrastructure tools, HIPAA requirements
- **AirBNB**: 5 Operations people for 1000+ instances
- **Gilt**: several hundred microservices, self service tools extending AWS, almost entirely on t2 instances
- **Intuit/TurboTax**: "deployed over 40 simultaneous experiments during the peak filing season"
- **MLB**: scale up massively during Minor League games and events, turn it off later



DevOps – Don't Take It Just From Us

In September 2015 Phil Calcado, ex-SoundCloud, wrote in "How we ended up with microservices." how SoundCloud reduced product development lead times from 66 days to 16!*




Where do you
START
?

Five major phases of software release

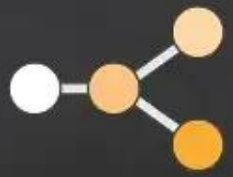
- Check-in source code such as .java files.
- Peer review new code

- Compile code
- Unit tests
- Style checkers
- Code metrics
- Create container images

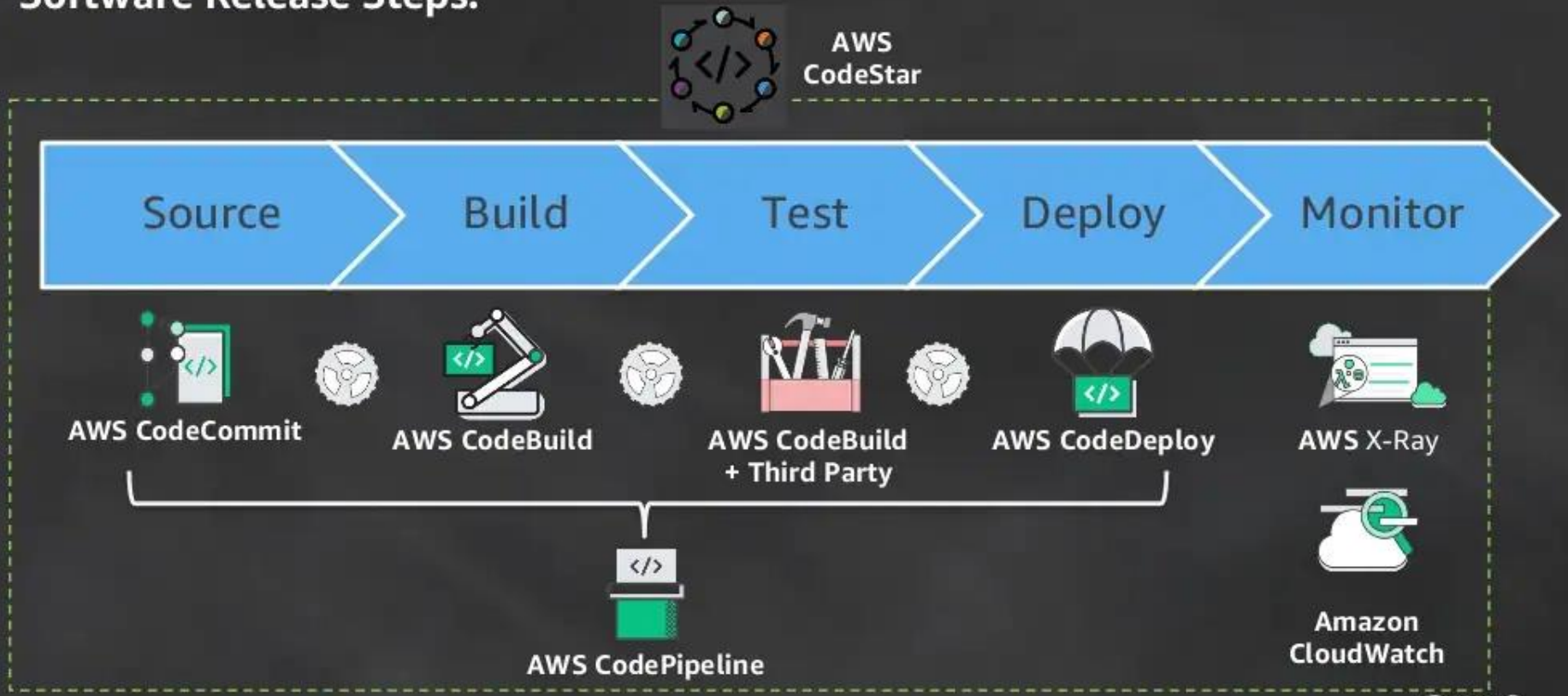
- Integration tests with other systems
- Load testing
- UI tests
- Penetration testing

- Deployment to production environments

- Monitor code in production to quickly detect unusual activity or errors



AWS Code Services
Software Release Steps:



AWS DevOps Portfolio

CI / CD Toolchain

AWS CodeCommit

AWS CodeStar

AWS CodeBuild

AWS CodeDeploy

AWS CodePipeline

Infrastructure as Code

AWS CloudFormation

AWS OpsWorks

AWS OpsWorks for Chef Automate

Monitoring & Logging

AWS X-Ray

Amazon CloudWatch

AWS CloudTrail

AWS Config

aws


Build & test your application
https://secure.flickr.com/photos/spenceyc/7481166880

https://www.flickr.com/photos/adamgiles/119754762

"This is our build server...

I mean, I think it is.
Someone else set it up.
They've left now.

Don't break it."

AWS CodeBuild

Fully managed build service that compiles source code, runs tests, and produces software packages

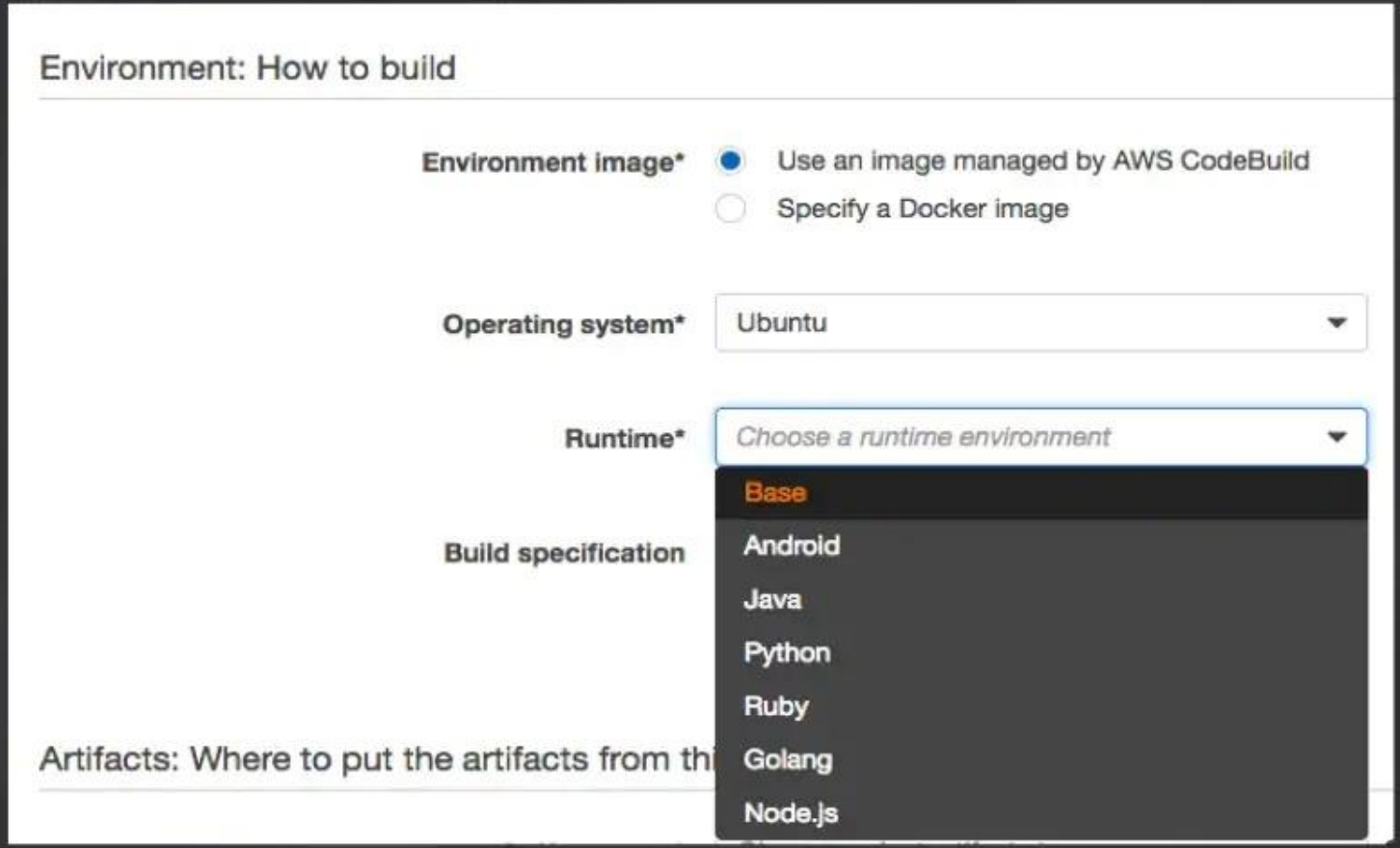
Scales continuously and processes multiple builds concurrently

You can provide **custom build environments** suited to your needs via Docker images

Pay by the minute for the compute resources you use

Integrated with CodePipeline and Jenkins

Configure a Build Project



buildspec.yml

```yaml
version: 0.1

phases:

  install:
    commands:
      - go get -u github.com/golang/lint/golint

  pre_build:
    commands:

      # Ensure code passes all lint tests
      - golint -set_exit_status

      # Run all tests included with our application
      - go test

  build:
    commands:

      # Build our application
      - go build -o app

artifacts:
  files:
    - app
```

buildspec.yml

- Sits in source repo alongside your project.
- Defines the commands to be run for each phase of the build, along with the output artifacts.
- Any errors will be reported back as a build failure, and the logs visible in the AWS CodeBuild console.

See build results

Phase details

	Name	Status	Duration	Completed
▸	SUBMITTED	Succeeded		Feb 25, 2017 12:04:11 AM UTC
▸	PROVISIONING	Succeeded	42 secs	Feb 25, 2017 12:04:54 AM UTC
▸	DOWNLOAD_SOURCE	Succeeded	4 secs	Feb 25, 2017 12:04:59 AM UTC
▸	INSTALL	Succeeded	21 secs	Feb 25, 2017 12:05:20 AM UTC
▸	PRE_BUILD	Succeeded	3 secs	Feb 25, 2017 12:05:23 AM UTC
▸	BUILD	Succeeded		Feb 25, 2017 12:05:24 AM UTC
▸	POST_BUILD	Succeeded		Feb 25, 2017 12:05:24 AM UTC
▸	UPLOAD_ARTIFACTS	Succeeded		Feb 25, 2017 12:05:25 AM UTC
▸	FINALIZING	Succeeded	5 secs	Feb 25, 2017 12:05:30 AM UTC
▸	COMPLETED	Succeeded		

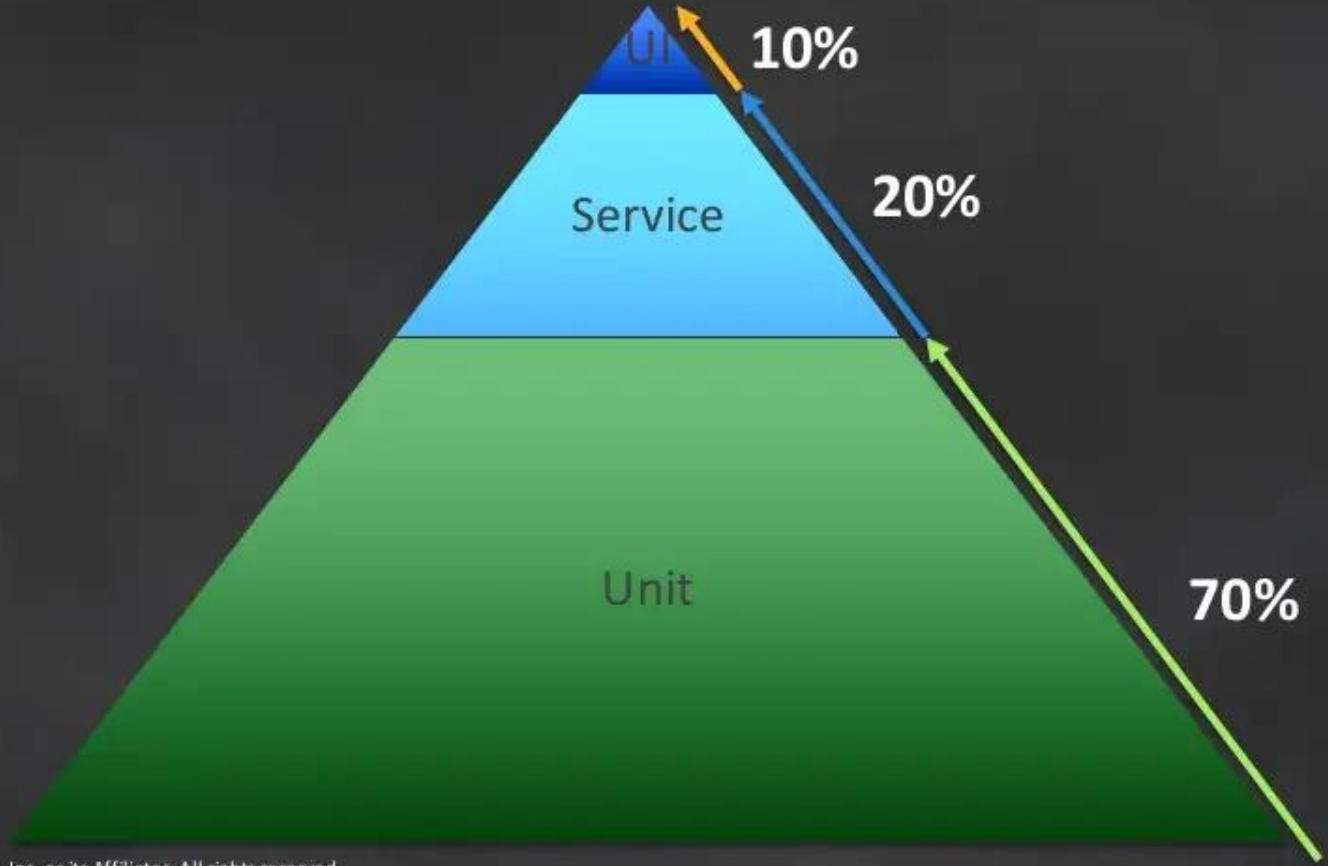
Build logs

Showing the last 20 lines of build log below. View entire log

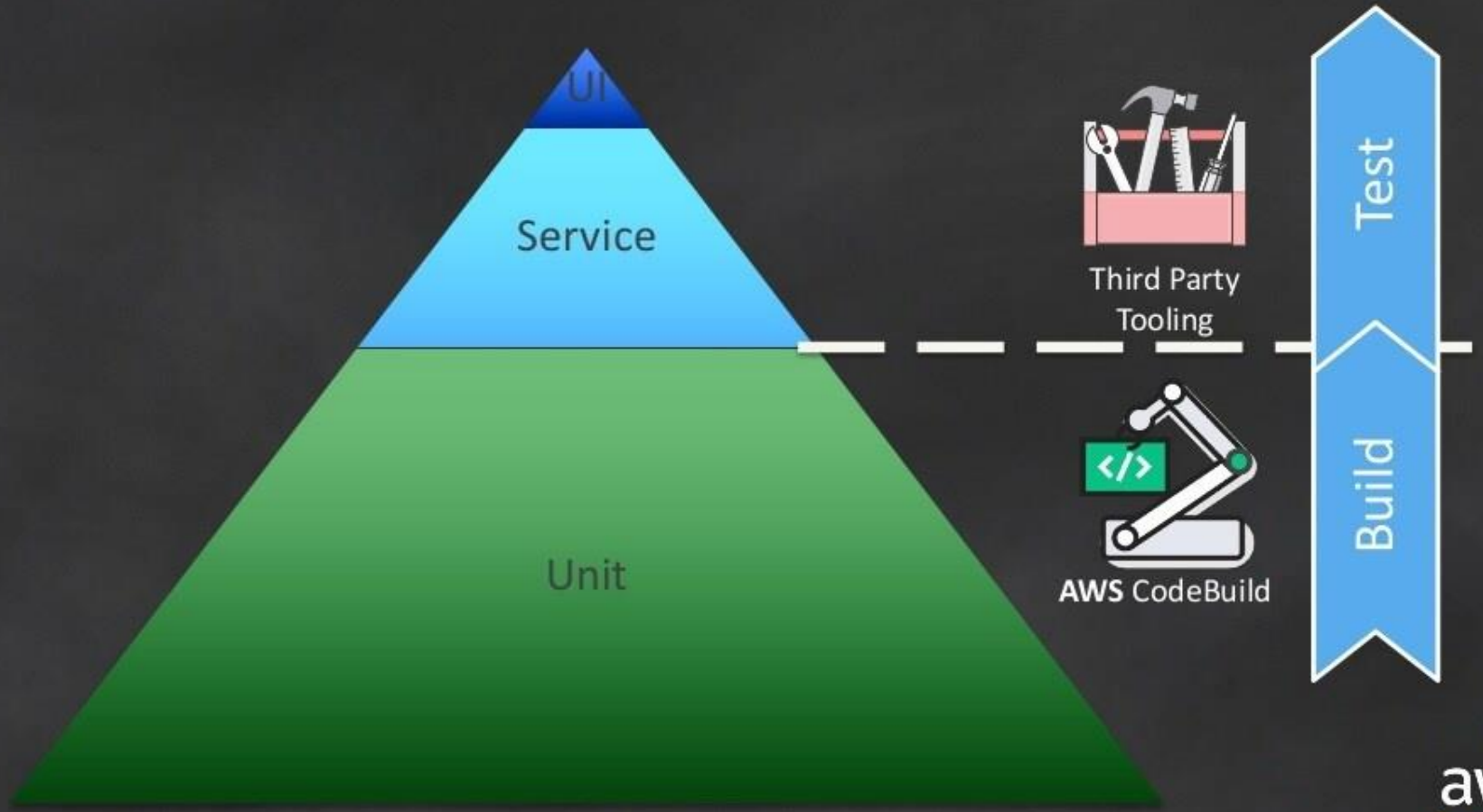
```
[Container] 2017/02/25 00:05:23 Phase context status code: Message:
[Container] 2017/02/25 00:05:23 Entering phase BUILD
[Container] 2017/02/25 00:05:23 Running command go build -o app
[Container] 2017/02/25 00:05:24 Phase complete: BUILD Success: true
[Container] 2017/02/25 00:05:24 Phase context status code: Message:
[Container] 2017/02/25 00:05:24 Preparing to copy artifacts
[Container] 2017/02/25 00:05:24 Expanding base directory path
[Container] 2017/02/25 00:05:24 Assembling file list
[Container] 2017/02/25 00:05:24 Expanding .
[Container] 2017/02/25 00:05:24 Expanding artifact file paths for base directory .
```

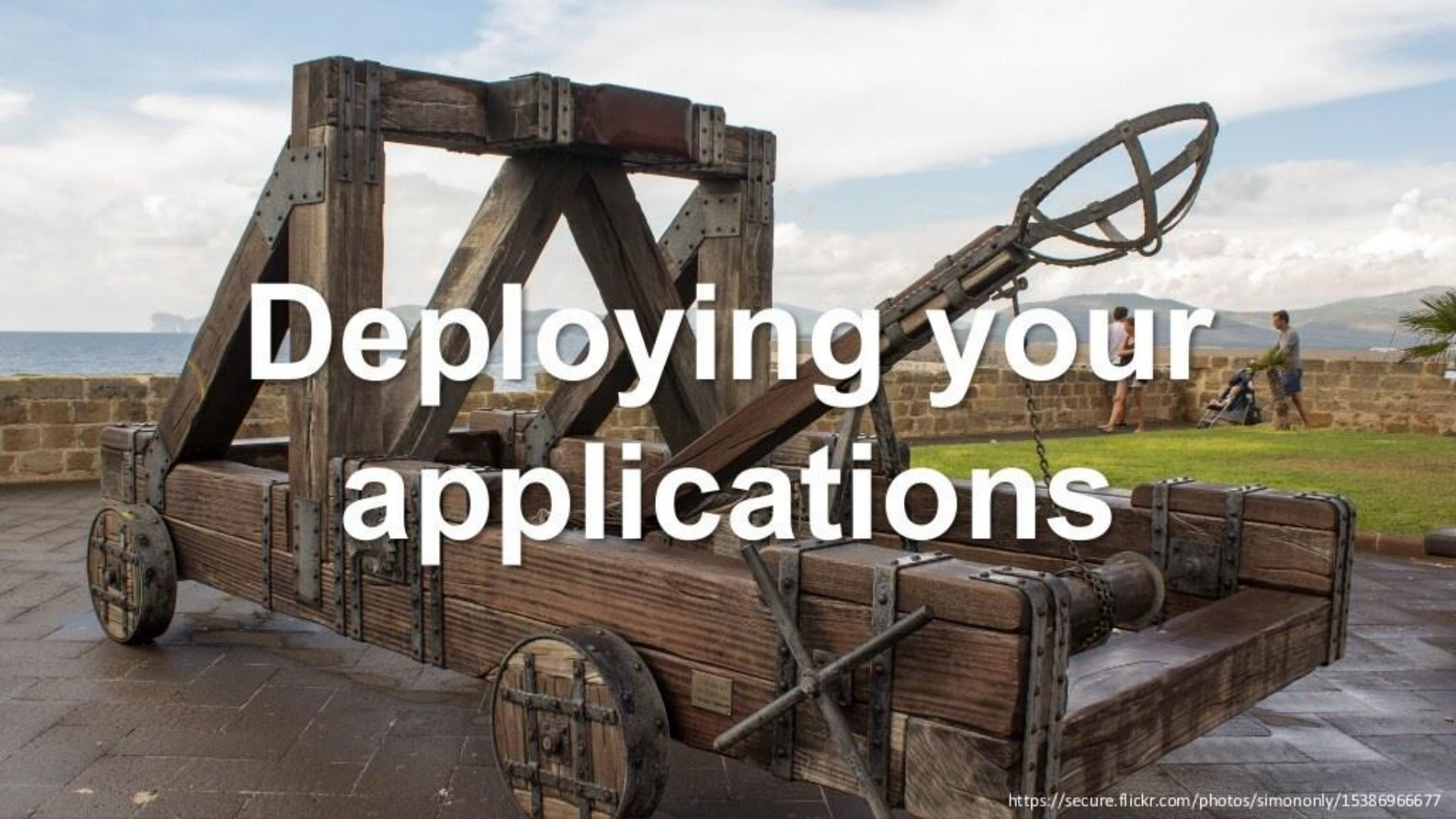

aws

Where to Focus Your Tests:



What service and release step corresponds with which tests?




Deploying your applications
https://secure.flickr.com/photos/simononly/15386966677

A measure of innovation agility

- **How many deployments am I performing?**

"We have a quarterly release cycle"

"Too many to count'"

- **How many are done out of hours?**

"We minimize customer impact"

"Time is irrelevant"

- **How many suffer emergency roll backs?**

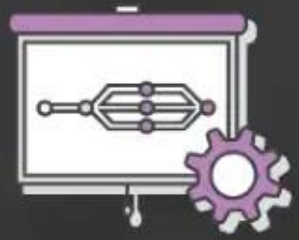
"We frequently catch problems too late and need to rollback from pre-release backups"

"We roll forwards not back"

aws

Strive for continuous deployment.
Use metrics and tooling to gain trust.

Continuous Delivery Benefits



Automate the software release process

Improve developer productivity

Find and address bugs quickly

Deliver updates faster

aws

AWS CodeDeploy

Automated deployments
Deploy to app running on AWS and/or On-premise

Minimize downtime
Supports rolling in-place deployments, as well as blue/green

Stop and roll back
You can automatically or manually stop and roll back deployments if there are errors.

Centralized control
You can launch and track the status of your deployments through the AWS CodeDeploy console or the AWS CLI. You will receive a report that lists when each application revision was deployed and to which Amazon EC2 instances.

Easy to adopt
Supports Windows and Linux. Works with any application. Also integrates with your CI/CD tooling or AWS CodePipeline.

appspec.yml Example

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: "*.html"
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  ApplicationStart:
    - location: scripts/start_httpd.sh
  ValidateService:
    - location: scripts/test_site.sh
      timeout: 60
  ValidateService:
    - location: scripts/register_with_elb.sh
```

- Send application files to one directory and configuration files to another
- Set specific permissions on specific directories & files
- Remove/add instance to ELB
- Install dependency packages
- Start Apache
- Confirm successful deploy
- More!

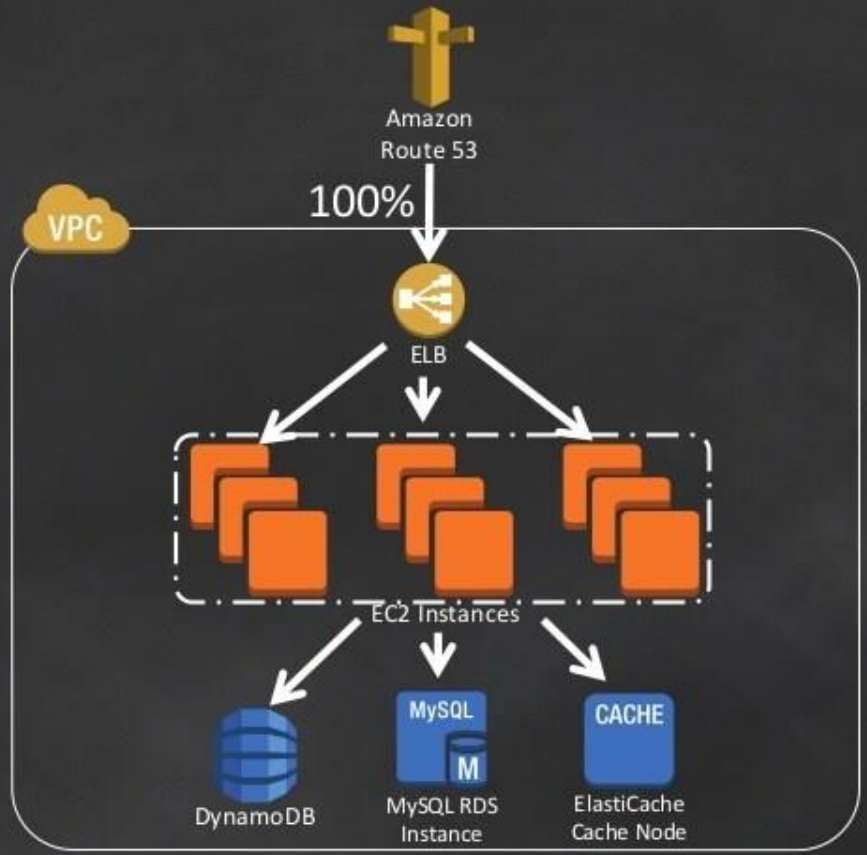
aws

Choose Deployment Speed and Group

AMI Deployment Method

Blue/Green Deploys

- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back
- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic

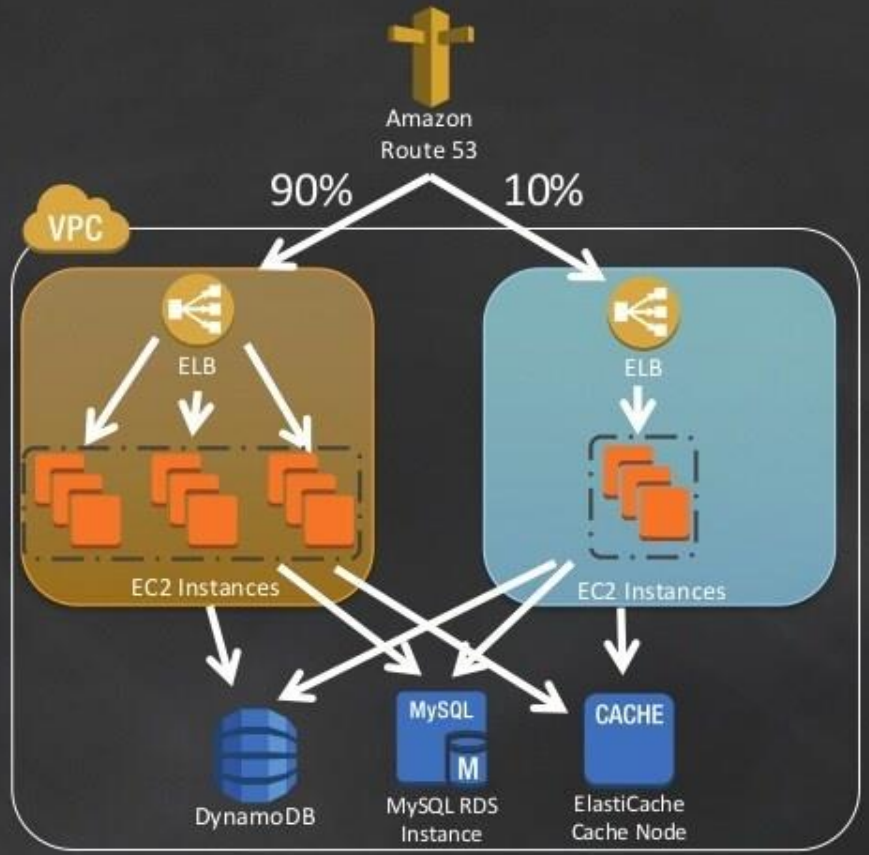


AMI Deployment Method

Blue/Green Deploys

- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back

- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic

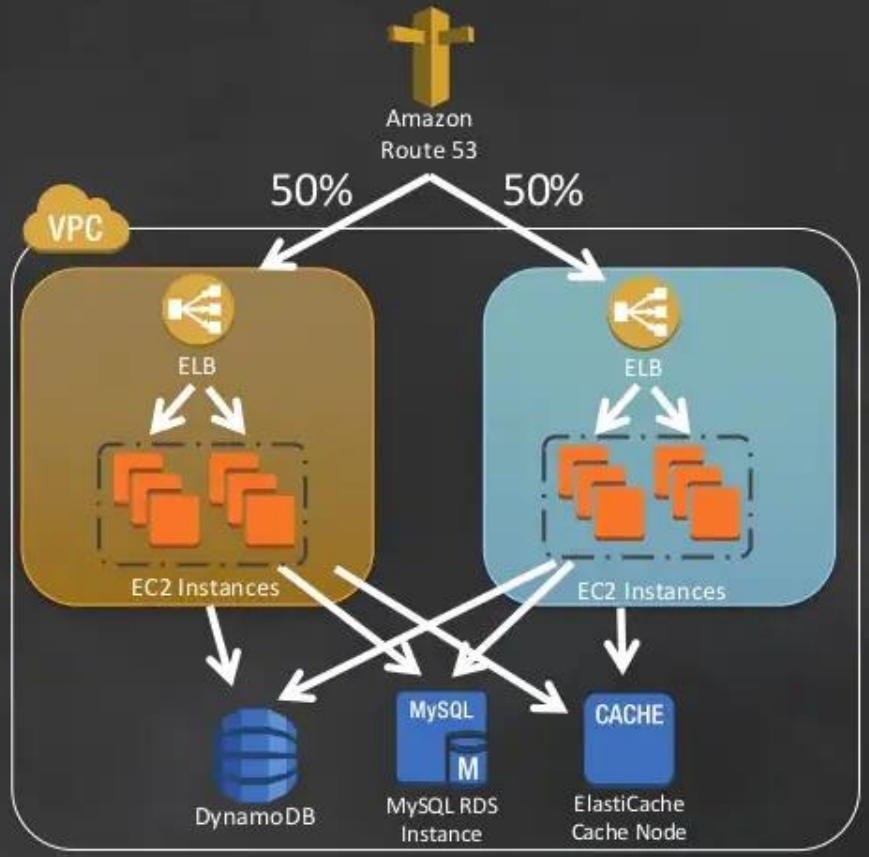


AMI Deployment Method

Blue/Green Deploys

- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back

- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic

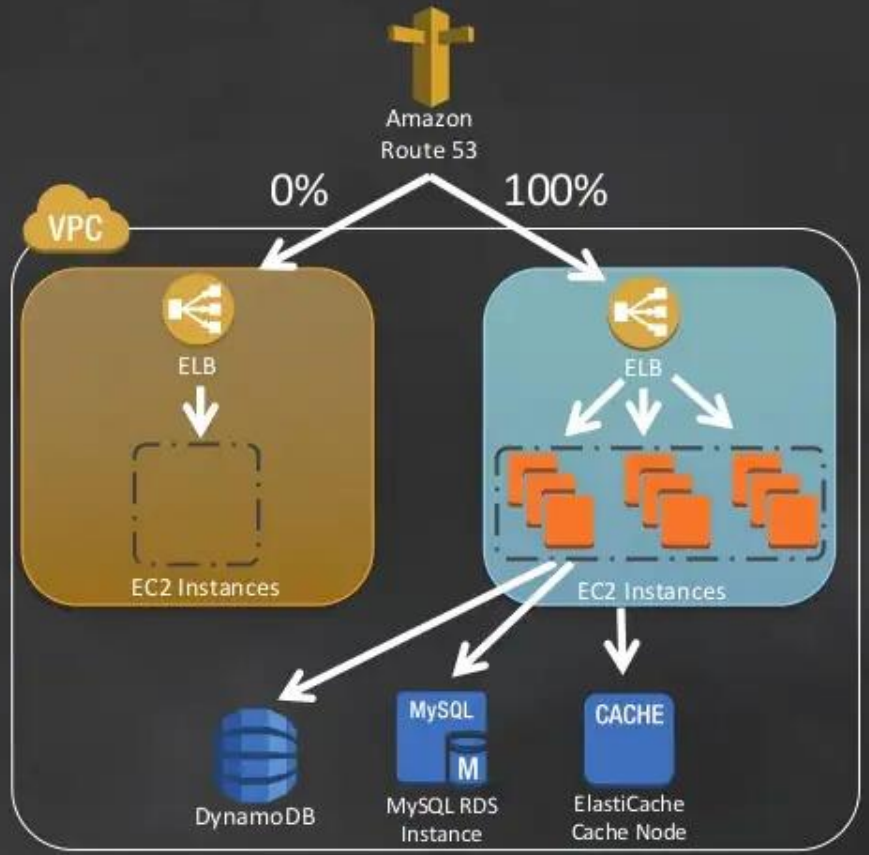


AMI Deployment Method

Blue/Green Deploys

- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back

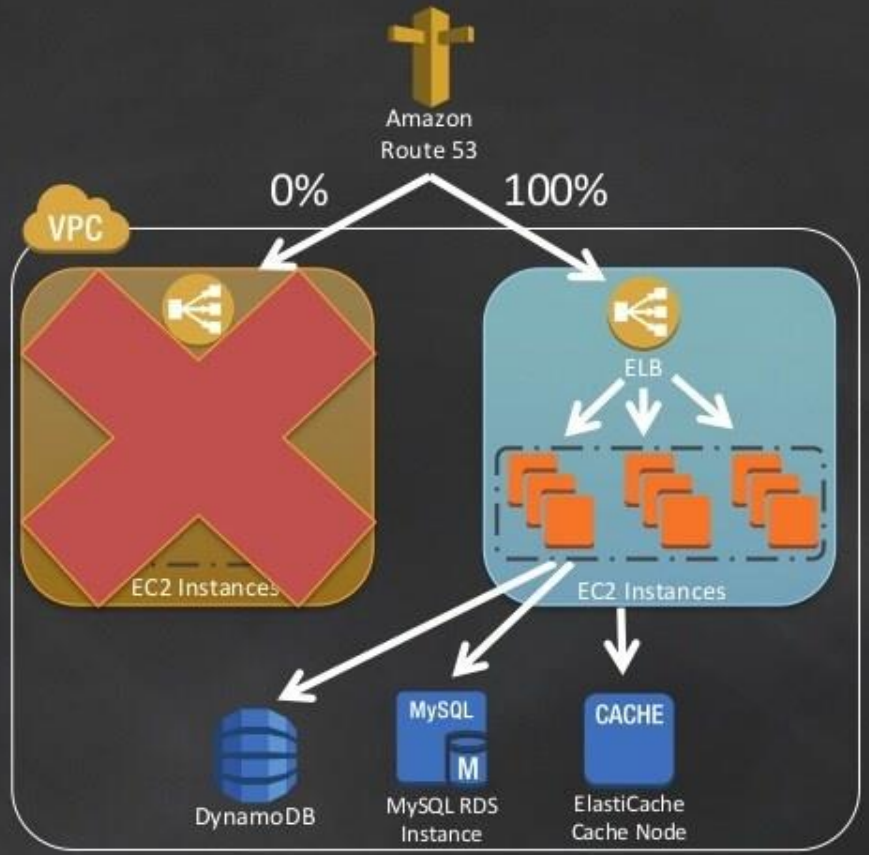
- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic



AMI Deployment Method

Blue/Green Deploys

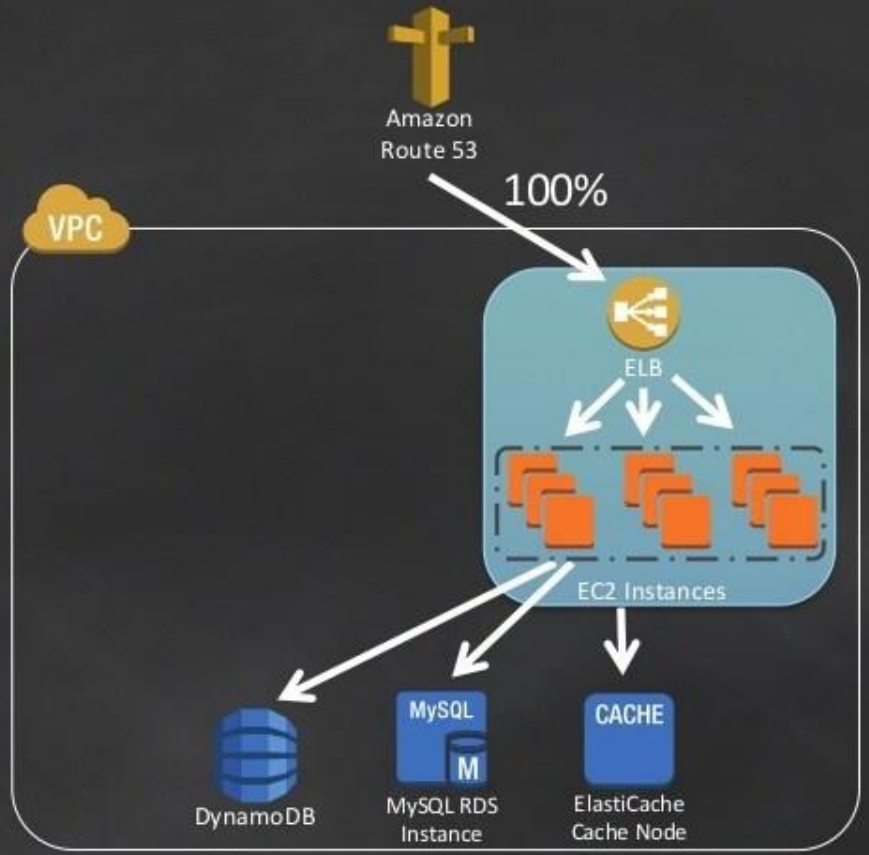
- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back
- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic



AMI Deployment Method

Blue/Green Deploys

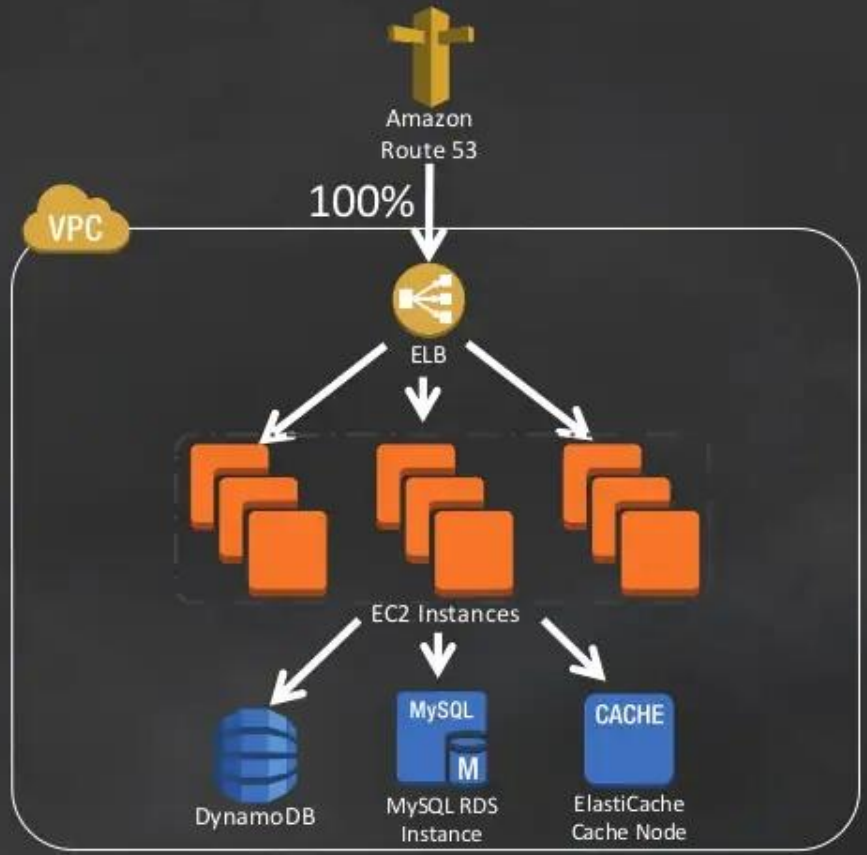
- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back
- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic



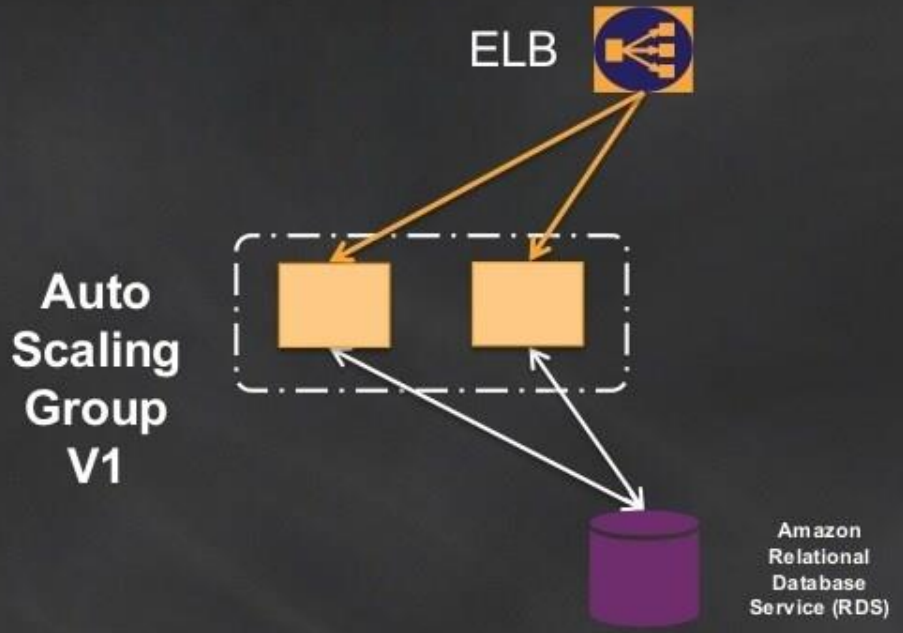
AMI Deployment Method

Blue/Green Deploys

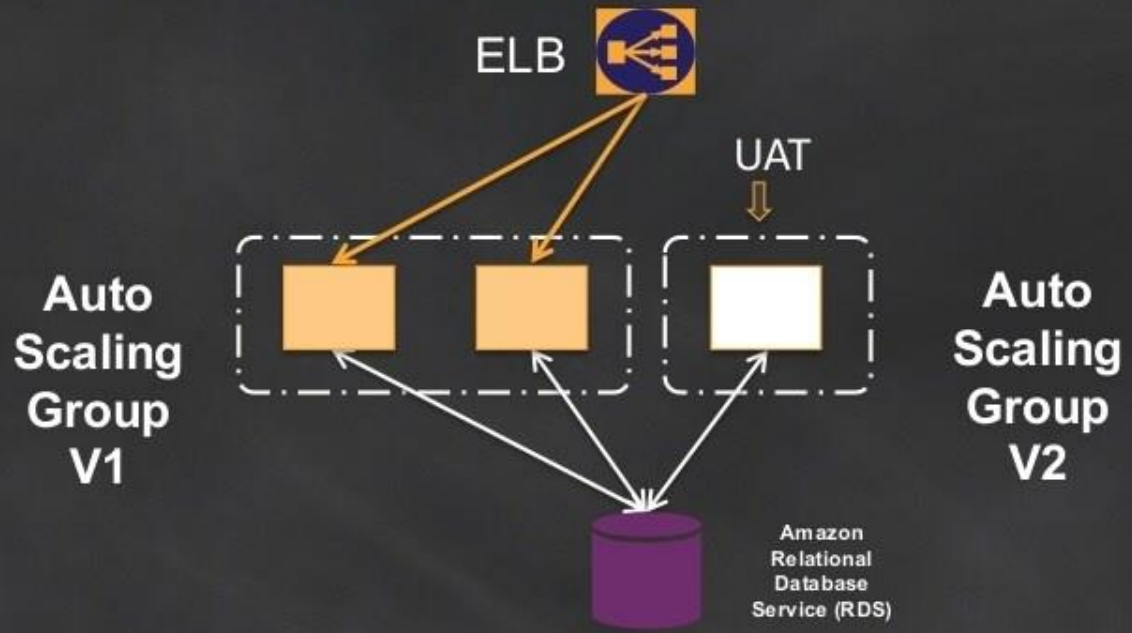
- We stand up a duplicate part of our infrastructure and slowly cut traffic over to it
  - ✓ Shift via DNS
  - ✓ Makes it easy to do testing of new features
  - ✓ Makes it easy to roll back
- As we shift more traffic over, let auto-scaling grow/shrink our instances of the new or old application
  - ✓ Shut down the old when no traffic



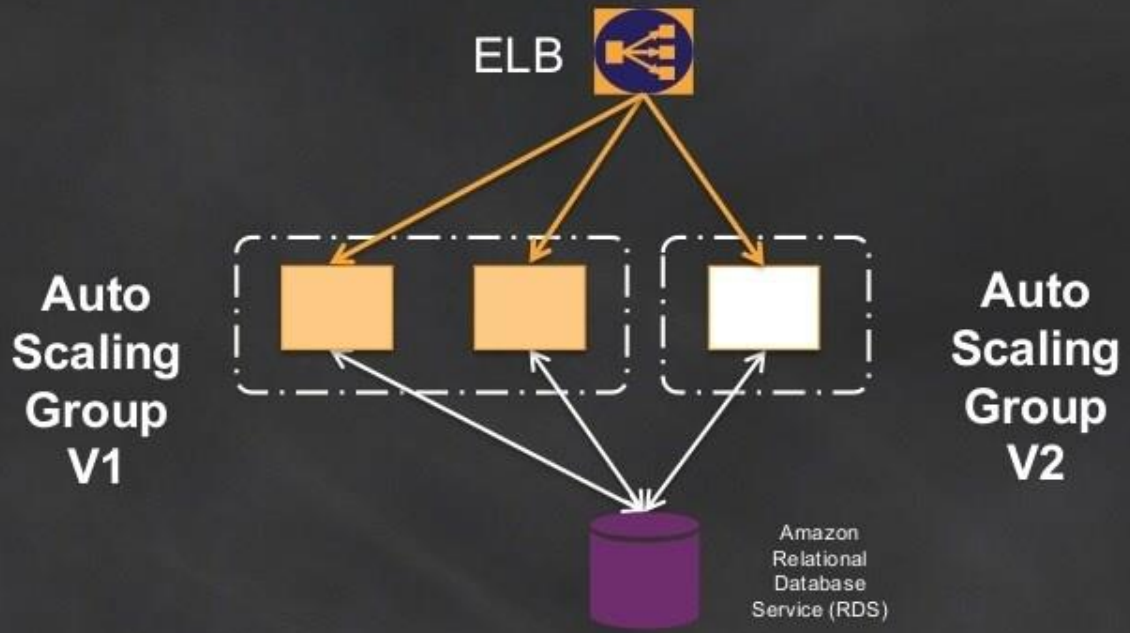
Red-Black Deployment



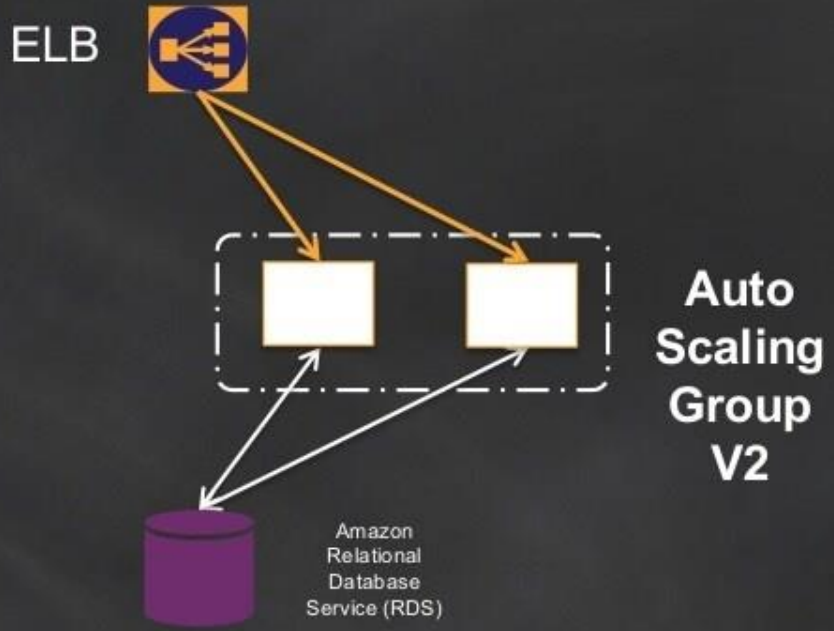
Red-Black Deployment



Red-Black Deployment

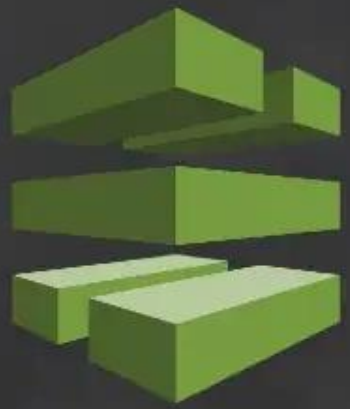


Red-Black Deployment




Orchestrating build and deploy with a pipeline
https://www.flickr.com/photos/seattlemunicipalarchives/12504672823/

AWS CodePipeline

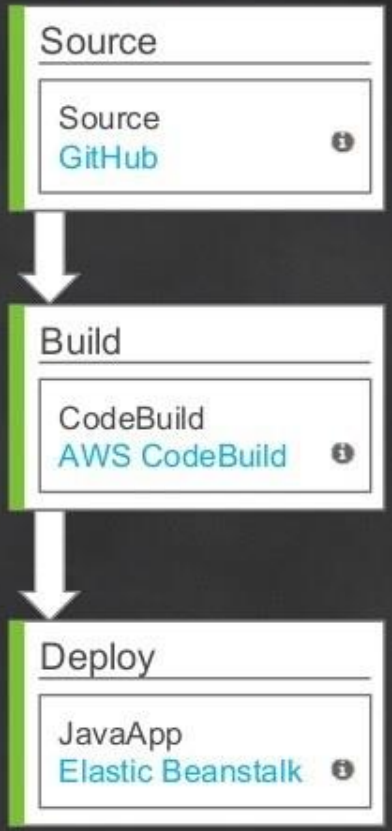


Continuous delivery service for fast and reliable application updates

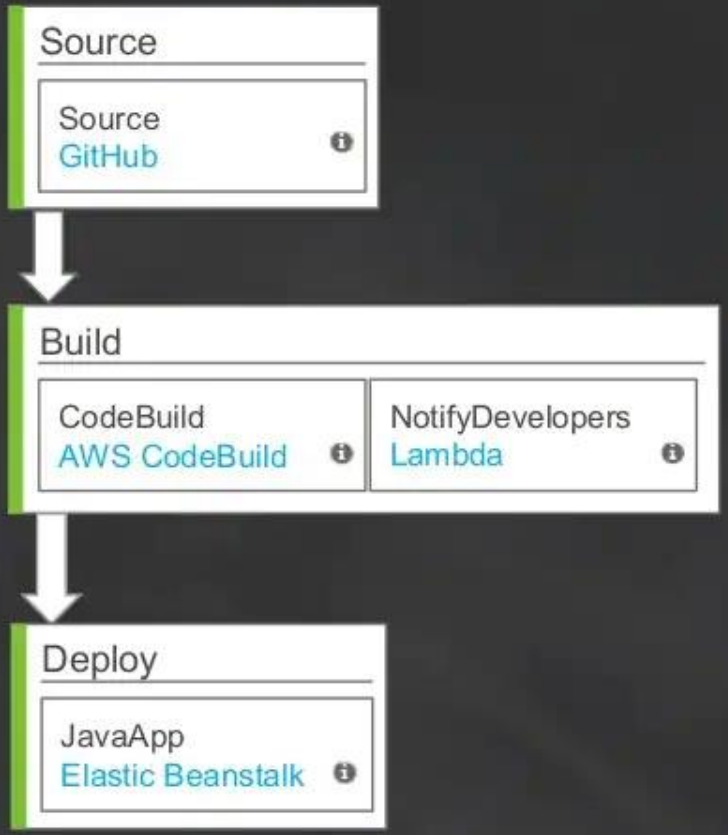
Model and visualize your software release process

Builds, tests, and deploys your code every time there is a code change

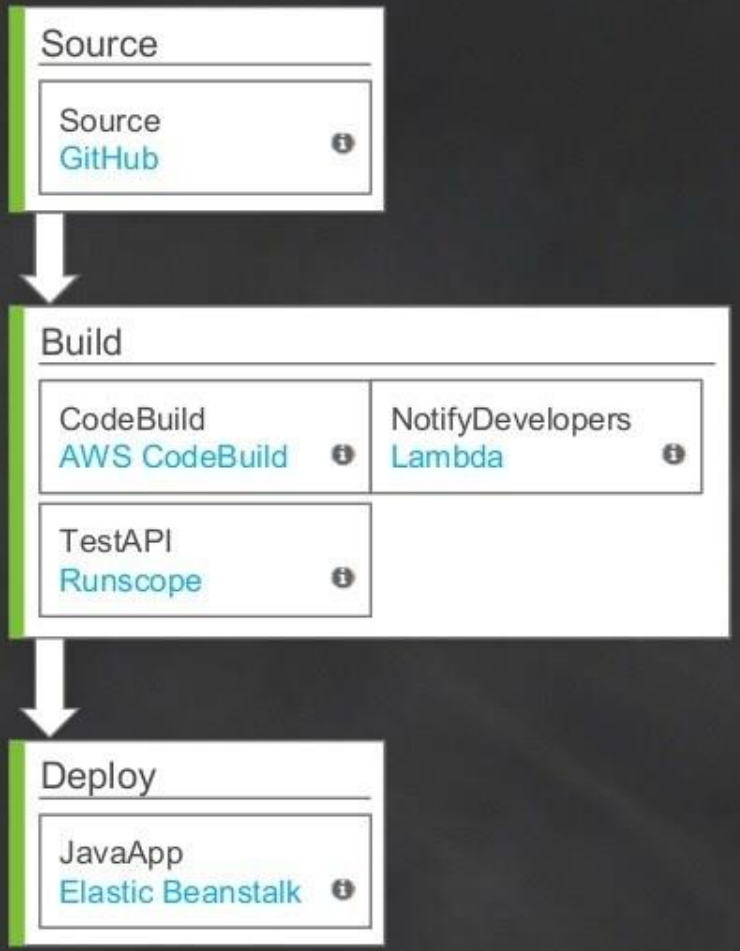
Integrates with third-party tools and AWS



AWS CodePipeline automatically picks up new source revisions from **AWS CodeCommit, GitHub or S3** and takes them through your build and release process.

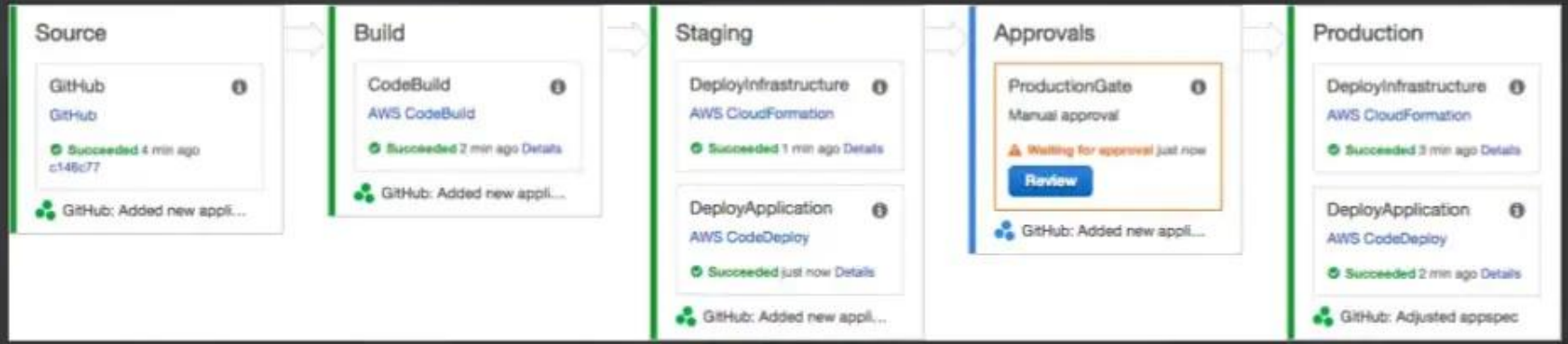


Parallel Actions



Sequential Actions

Putting it all together…



Summary

- **Innovation requires agility**. Teams busy firefighting, and large releases are the enemies of agility
- Shrink Deployments and strive for **continuous delivery**
- **Improve visibility** with AWS CodePipeline
- Automate and **scale your builds** with AWS CodeBuild
- Implement **safe, zero-downtime deployments** with AWS CodeDeploy
- Choose your ideal **deployment strategy**.

aws

Enjoy developing your software!