DevOps at Amazon

Table of contents

Amazon’s story

The “not whiteboard” edition

The three laws of Amazon DevOps**

1.

Break
things down

2.

Teams are
autonomous
businesses

3.

Automate
everything

**Strictly according to me

The three* laws of Amazon DevOps**

1.

Break
things down

2.

Teams are
autonomous businesses

3.

Automate
everything

4.

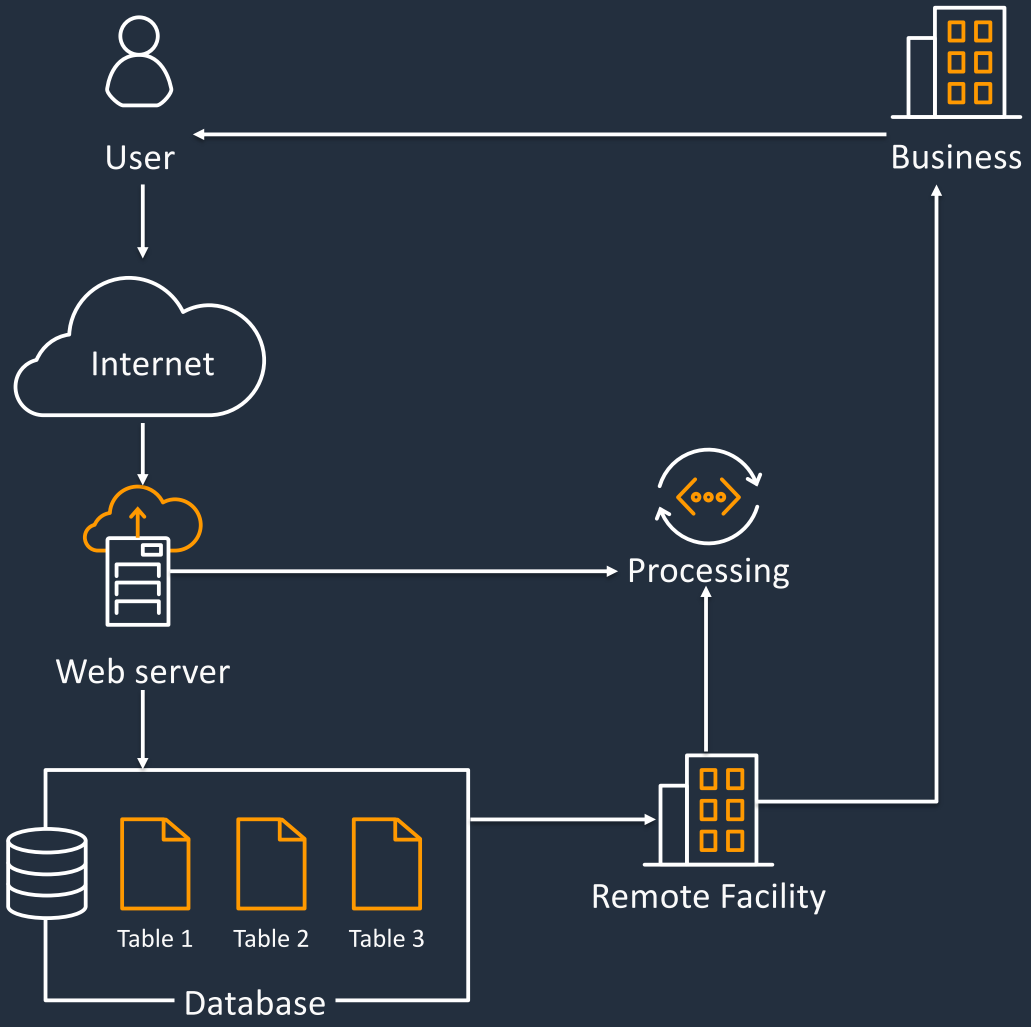
*Manage the
things that matter

**Strictly according to me

Starting out

This is how many web architectures started out, and it's how Amazon started too…

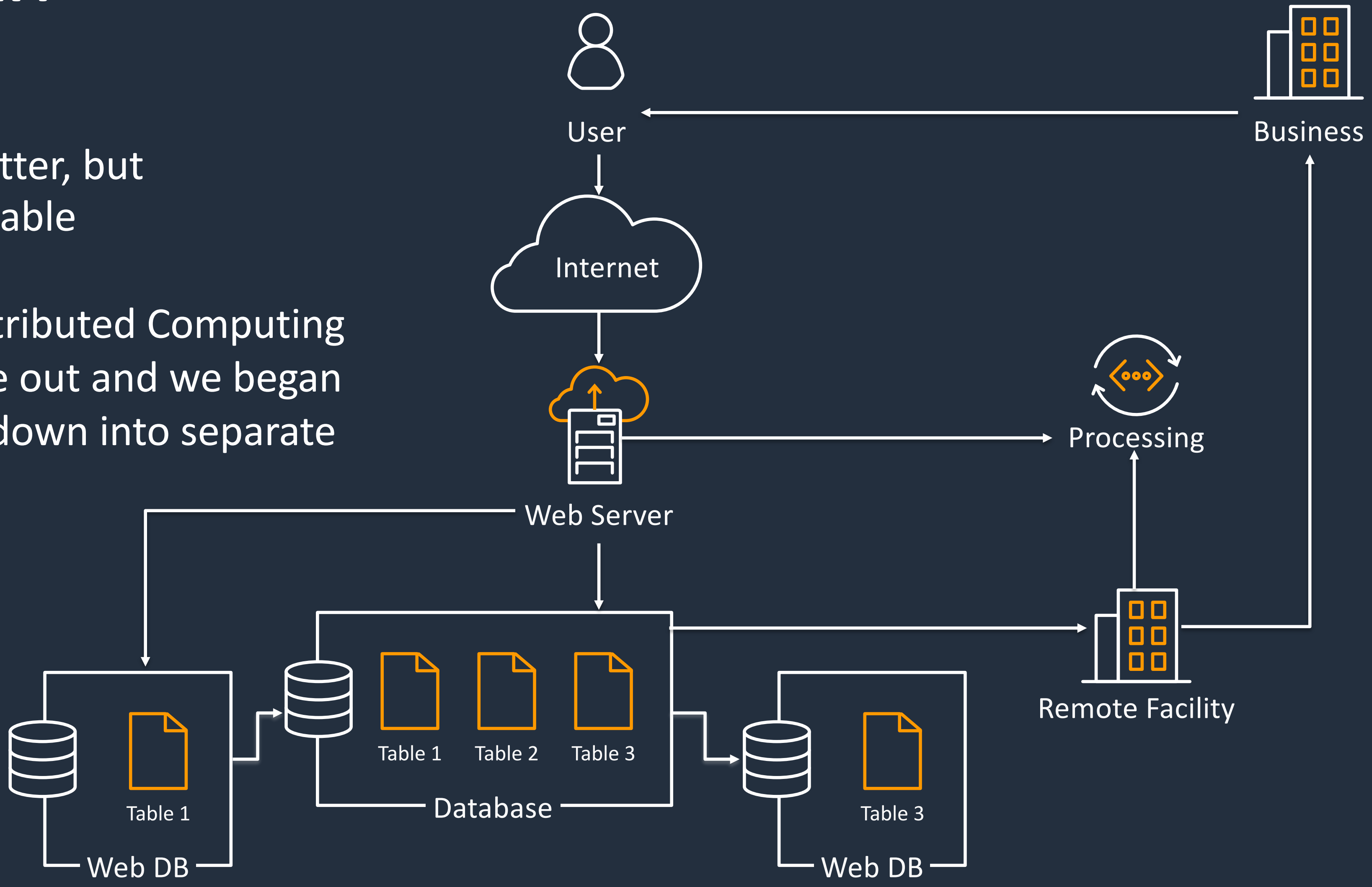
There any many bottlenecks, and scaling of the web server can was an immediate factor



Scaling Mark I

This was a bit better, but still not very scalable

In 1998 the "Distributed Computing Manifesto" came out and we began breaking things down into separate components…

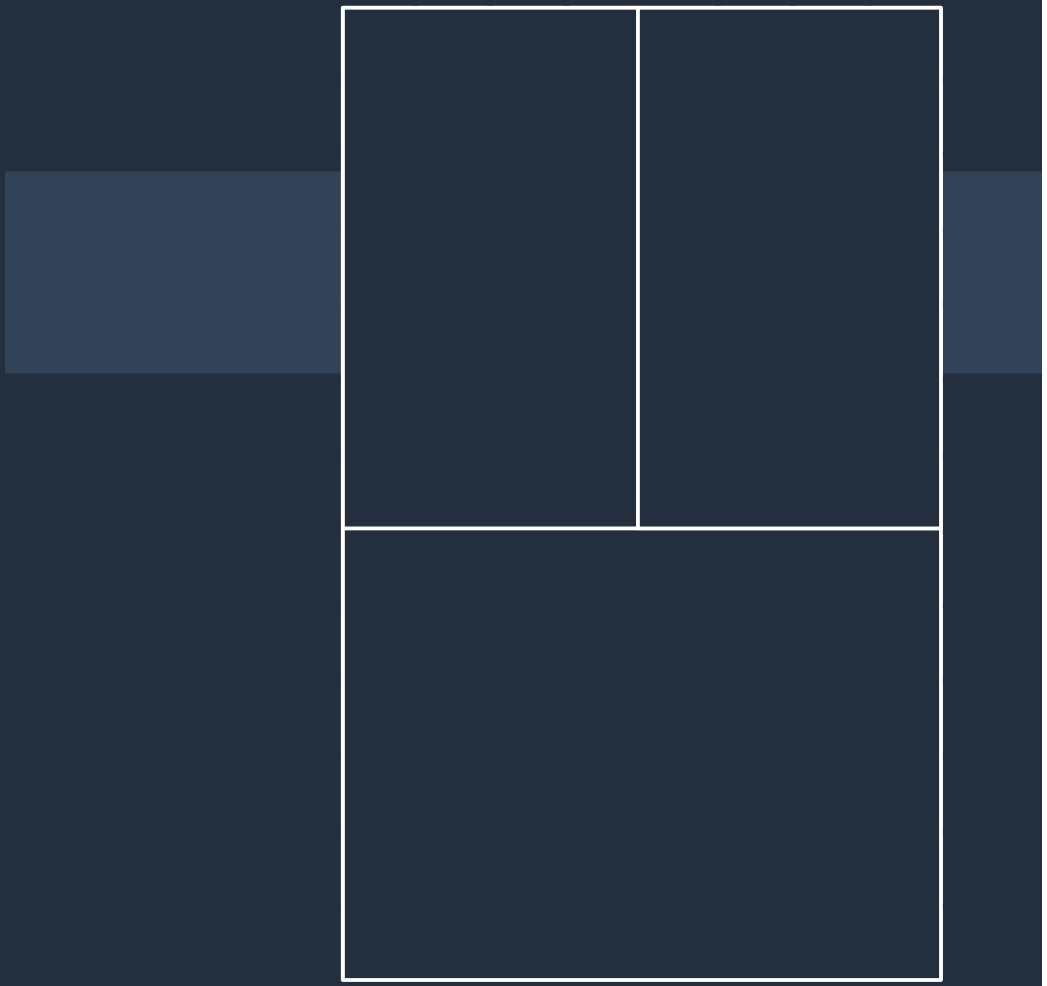


Breaking things down

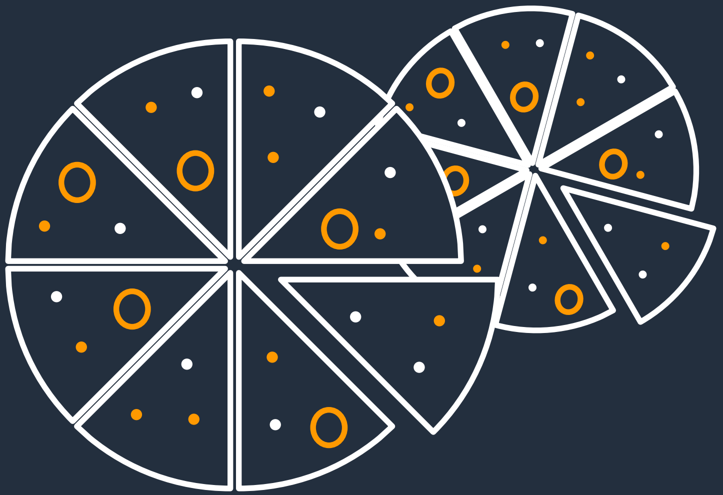
Principles

- Make units a small as possible (Primitives)
- Create data domains
- De-couple based on scaling factors, not functions
- Each service operates independently
  “Communication is terrible!” —Jeff Bezos
- APIs (contracts) between services

✓ **This led to changes in organization**



Getting (re)organized



“Two-pizza” teams

- Own a service
- Minimizes social constraints (Conway’s law)
- Autonomy to make decisions

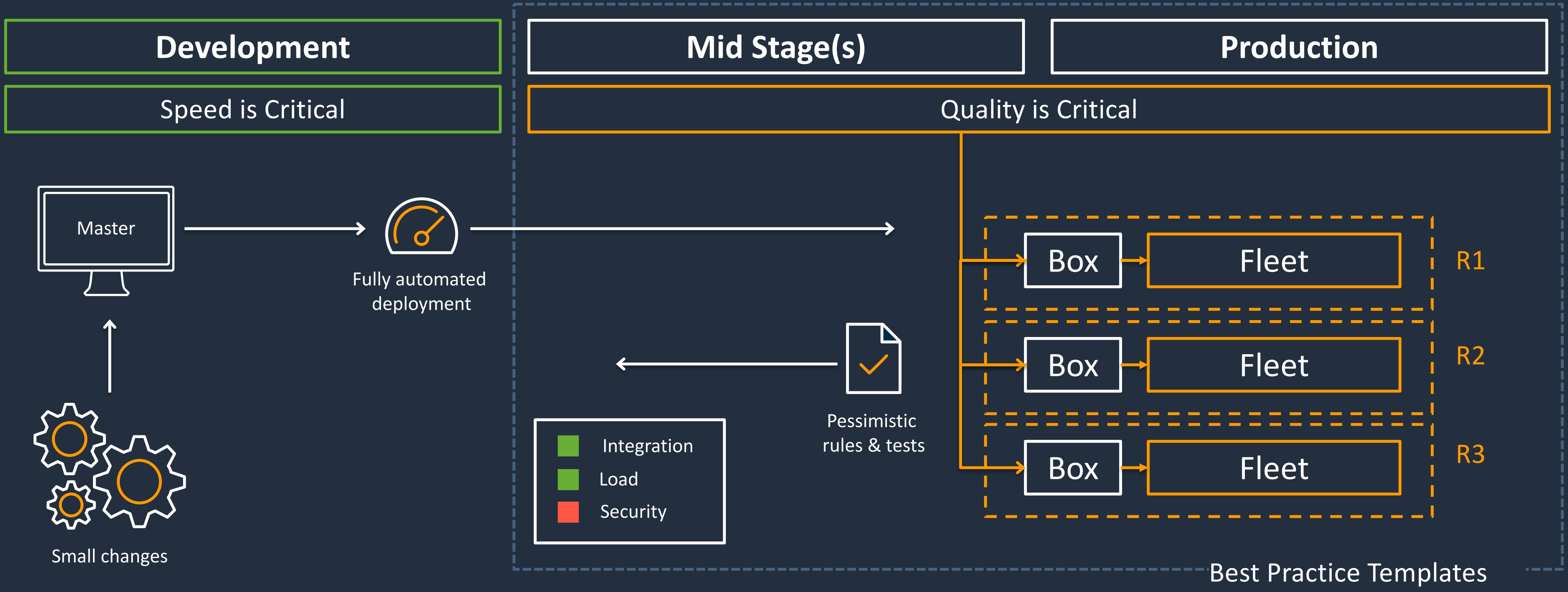
Getting (re)organized

Own everything

- Planning
- Security
- Performance
- Scalability
- Deployment
- Operation
- Bugs
- Documentation
- Testing…

How can one team do all of this?

Automate everything



Managing success

Business metrics	Operational metrics	Input goals	Enablement
Growth	Errors	Features	Principal reviews
Usage	Throttling	Use cases	Security training
Feedback	Failed deployments	Performance	Ops training
	Performance	Features	

Today we have Modern Applications

Modern Application

- Does one thing
- Independent deployments
- Independent scaling
- Small impact of change
- Choice of technology

Today we have Modern Applications

Modern Applications

- Use independently scalable microservices (serverless, containers...)
- Connect through APIs
- Deliver updates continuously
- Adapt quickly to change
- Scale globally
- Are fault tolerant
- Carefully mange state and persistence
- Have security built-in

aws

Applies across industries

oscar	airbnb	GILT	intuit turbotax	
2 systems engineers 45+ developers Self-service Infrastructure tools HIPAA requirements	5 Operations people for 1000+ instances	Several hundred micro services Self service tools extending AWS Almost entirely on t2 instances	"Deployed over 40 simultaneous experiments during the peak filing season"	Scale up massively during Minor League games and events, turn it off later

Strong ecosystem

ATLASSIAN

Apica

BlazeMeter

XL XebiaLabs
Enterprise DevOps

Travis CI

GitHub

cloudbees

salesforce heroku

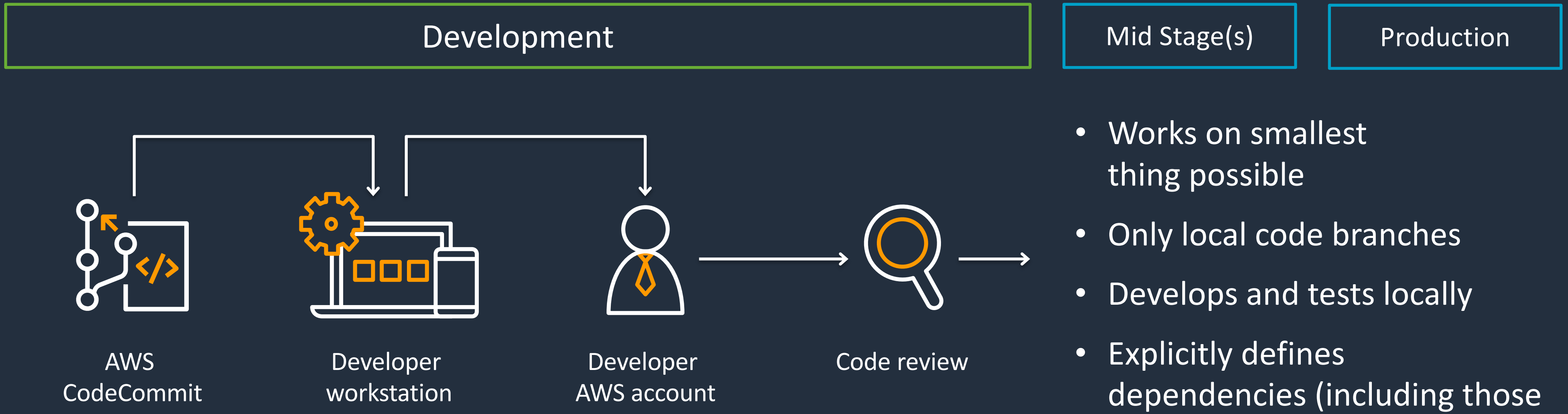
Layered Insight™

CODESHIP
by CloudBees

aws

Going deeper…

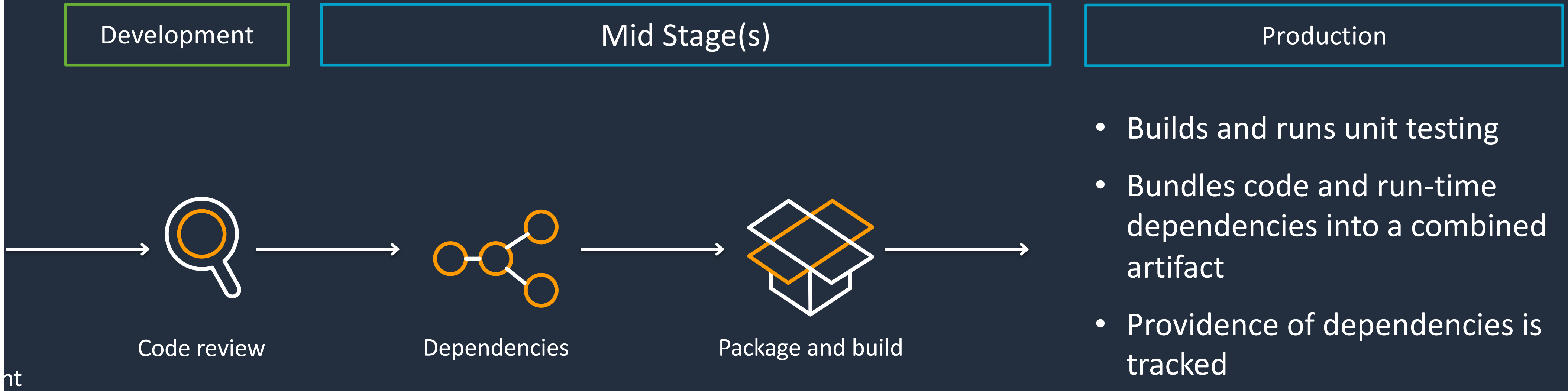
Example pipeline



- Works on smallest thing possible
- Only local code branches
- Develops and tests locally
- Explicitly defines dependencies (including those needed for testing)
- Publishes changes for review

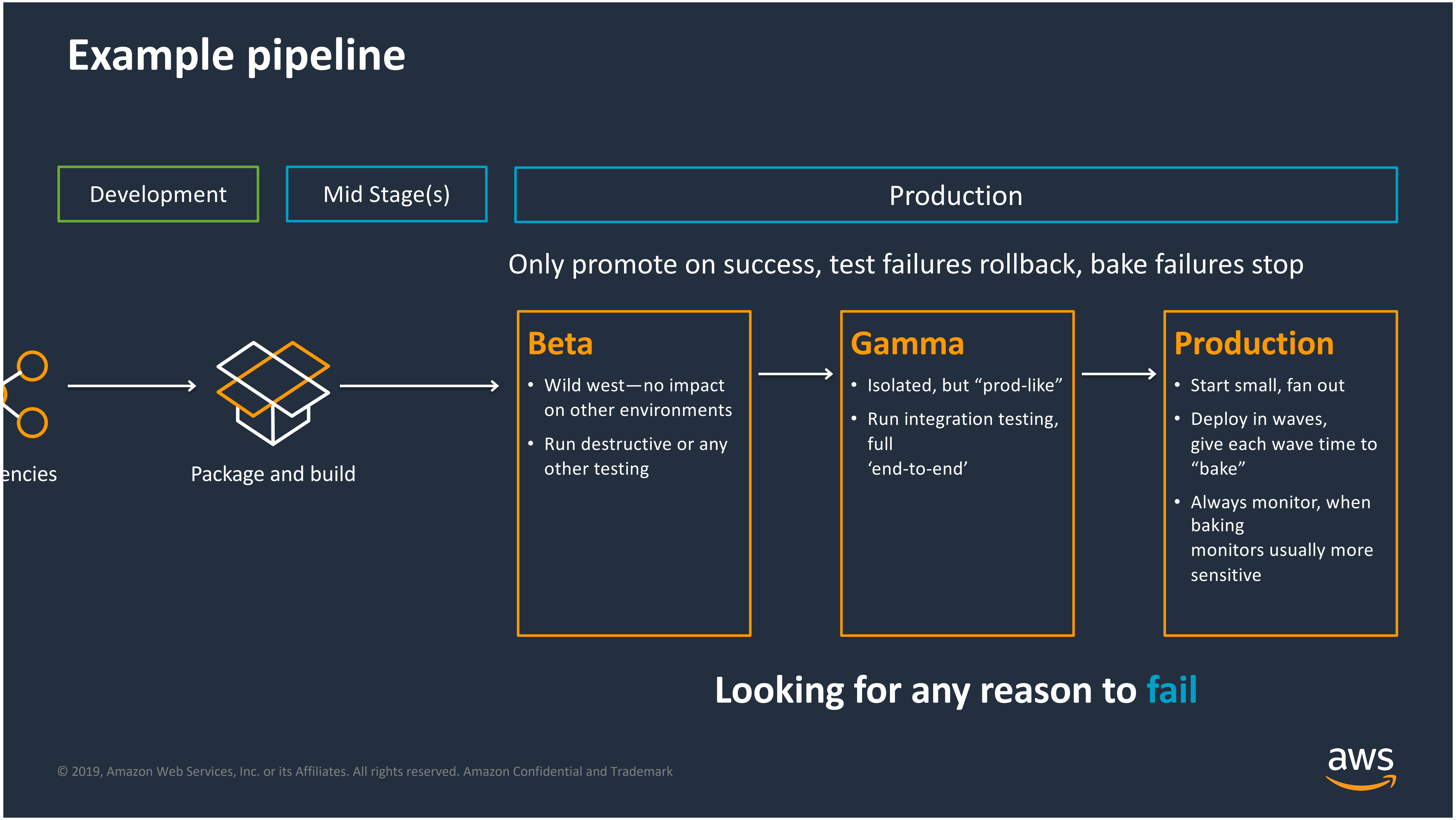
Should be fast

Example pipeline



- Builds and runs unit testing
- Bundles code and run-time dependencies into a combined artifact
- Providence of dependencies is tracked

Should be thorough



Example pipeline

Development

Mid Stage(s)

Production

Only promote on success, test failures rollback, bake failures stop

encies

Package and build

Beta

- Wild west—no impact on other environments
- Run destructive or any other testing

Gamma

- Isolated, but “prod-like”
- Run integration testing, full ‘end-to-end’

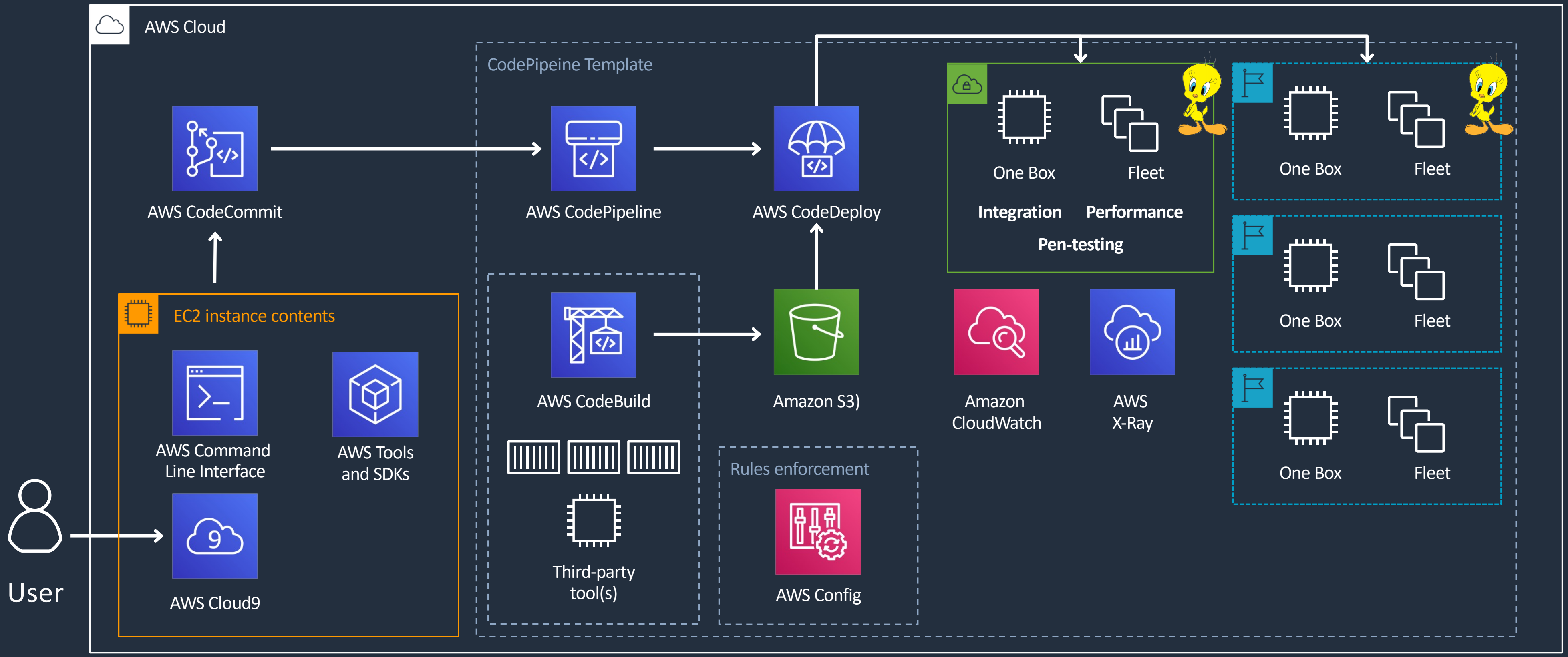
Production

- Start small, fan out
- Deploy in waves, give each wave time to “bake”
- Always monitor, when baking monitors usually more sensitive

Looking for any reason to fail

aws

Example pipeline architecture



Q&A

Additional content

AWS DevOps Workshop

https://s3.amazonaws.com/aws-devops-workshop/site/index.html

Integrating Git with AWS CodePipeline

https://s3.amazonaws.com/aws-devops-workshop/site/index.html

AWS CodeStar

https://aws.amazon.com/codestar

AWS CodeBuild

https://aws.amazon.com/codebuild

AWS CodePipeline

https://aws.amazon.com/codepipeline

AWS CodeCommit

https://aws.amazon.com/codecommit

AWS CodeDeploy

https://aws.amazon.com/codedeploy

Plus many YouTube videos from re:invent 2018, 2017…and Twitch AWS programming year-long

Thank you!