

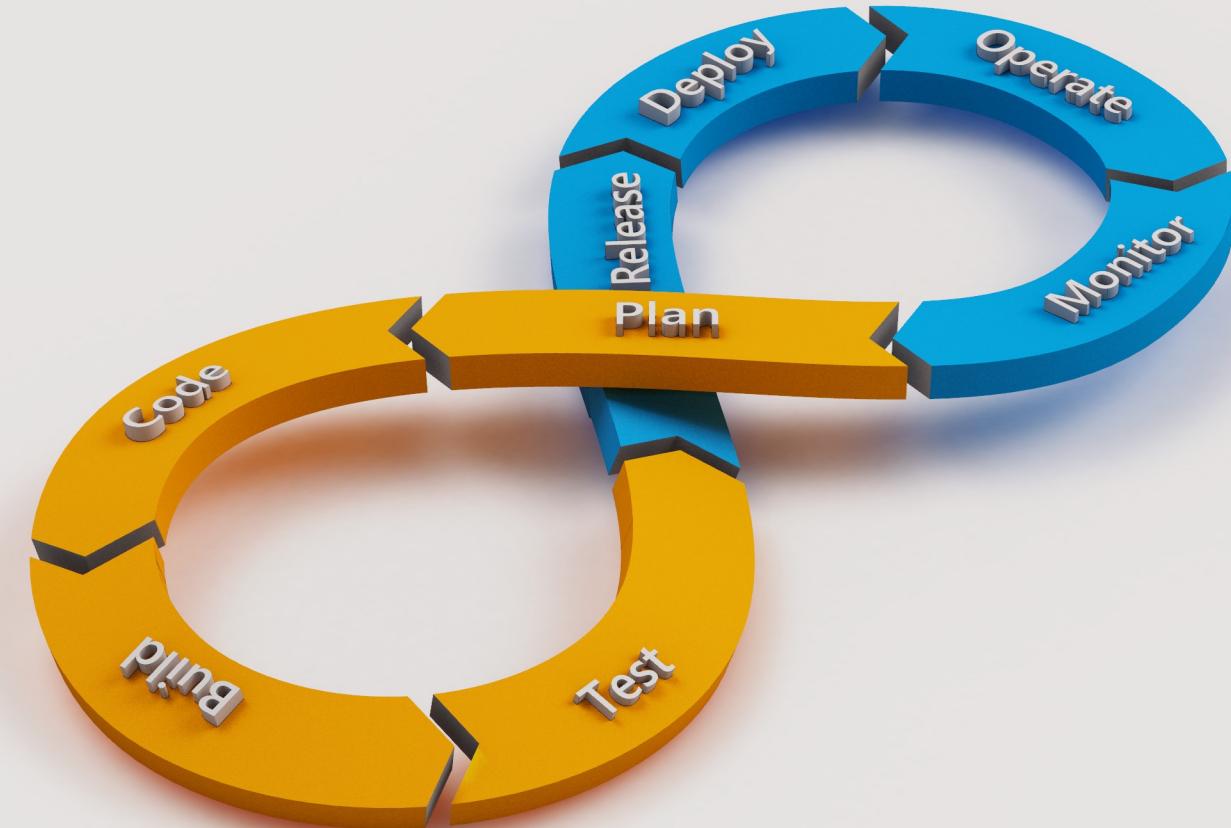
# CI/CD In the Serverless Era

Efi Merdler-Kravitz

Serverless Hero, VP of Engineering, Lumigo

✉️ [efi@lumigo.io](mailto:efi@lumigo.io)

🐦 [@TServerless](https://twitter.com/TServerless)





# BUT WHAT'S IN IT FOR ME?

- 100 % Serverless in production
- 100 % Serverless in CI/CD
- Tips to speed the flow and make it more robust
- Unique flows for phased deployment

# Some stats

- ~ 20K individual test runs per month
- ~ 30 deploys to production per week
- ~ 1H average deploy time to integration environment
- ~ 40m average time to run the tests

# Who am I



AWS Serverless Hero  
Head of Lumigo R&D  
3<sup>rd</sup> company I'm using Serverless



Working at Lumigo in the past three years:

- SaaS platform for AWS serverless observability
- GA since Q1'2019

# Agenda

- The DevOps Infinity Loop - Revisited
- Our best practices



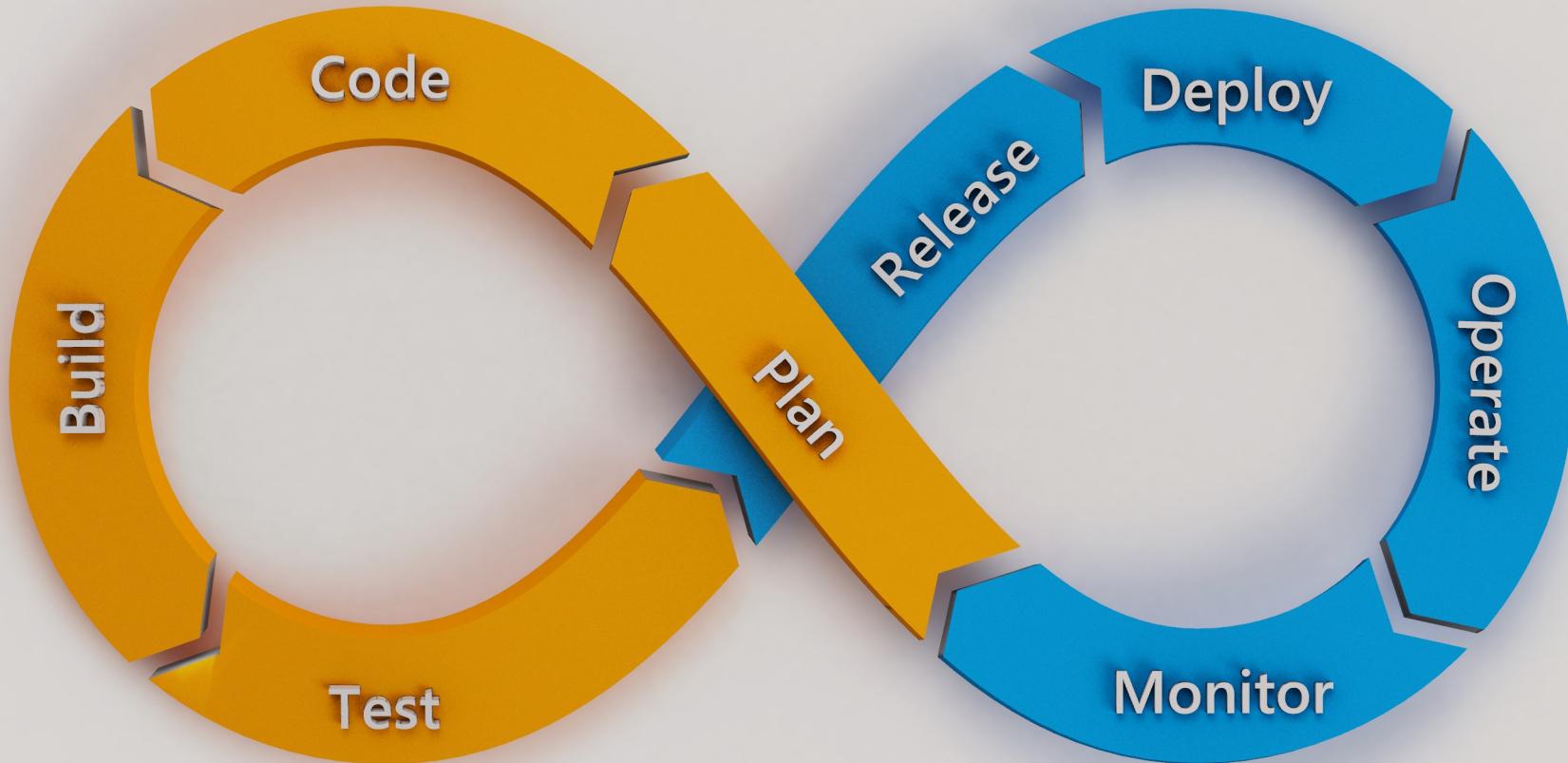
# Serverless is Different



## DISTRIBUTED MICROSERVICES

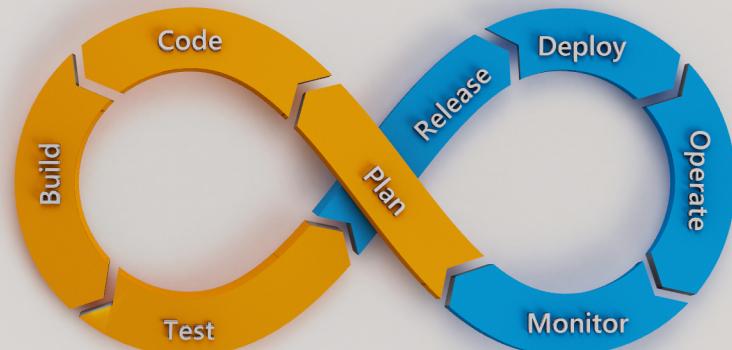
- Hundreds of components
  - Can't run locally
  - Frequent deployments

# The Infinity Loop



# Guidelines

- Own environment
- Serverless services
- No dedicated QA or Ops



# Environments



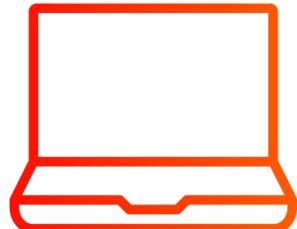
Personal  
Environment



Integration  
Environment



Production

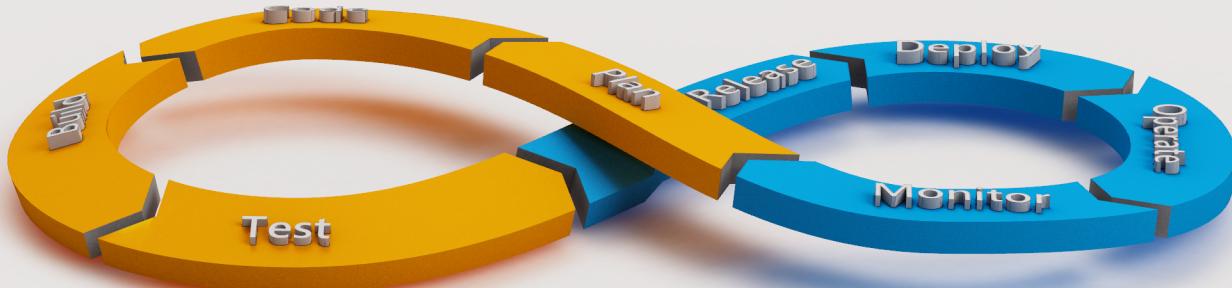


Local  
Development



Monitoring  
Environment

# Our Stack



<https://tinyurl.com/proactive-lumigo>

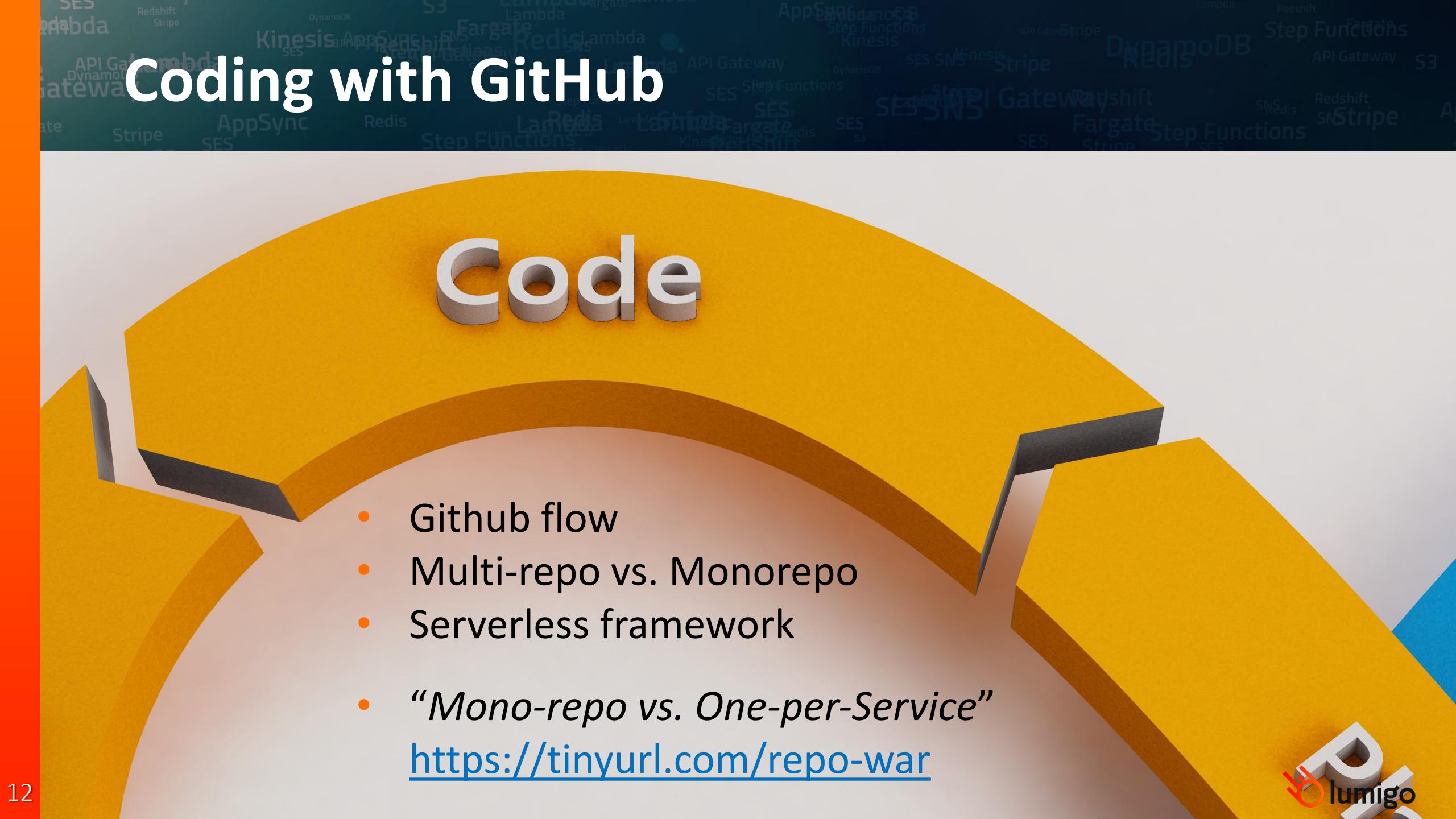


# Planning with JIRA

## Plan

- Kanban → Scrum
- JIRA
  - [+] Industry standard
  - [+] Integrations (GitHub)
  - [-] Can be slow/complex

# Coding with GitHub



# Code

- Github flow
- Multi-repo vs. Monorepo
- Serverless framework
- “*Mono-repo vs. One-per-Service*”  
<https://tinyurl.com/repo-war>

# Coding

- YML per service
- Uber deploy gets the needed code

```
Shays-MacBook-Pro:sls_deploy shay$ python3 main.py --env demo
Working directory: /Users/shay/source
Clone & pull git repositories: data-access, lumigo-tracer, python_tracer, integration-tests, python-common-utils, common-resources, log-shipping, tracing-ingestion, resources-details, system-map, logs, alerting, java-tracer

data-access      : success! (pulled master)
lumigo-tracer   : success! (pulled master)
python_tracer   : success! (pulled master)
integration-tests: success! (pulled master)
python-common-utils: success! (pulled master)
common-resources: success! (pulled master)
log-shipping     : success! (pulled master)
tracing-ingestion: success! (pulled master)
resources-details: success! (pulled master)
system-map       : success! (pulled master)
tracing-ingestion-edge: success! (pulled master)
logs            : success! (pulled master)
alerting         : success! (pulled master)
java-tracer     : success! (pulled master)

Init venv & dependencies for python repositories: data-access, python_tracer, alerting, system-map, common-resources, log-shipping, logs, resources-details, tracing-ingestion, tracing-ingestion-edge
.....
data-access      : success!
python_tracer   : success!
alerting         : success!
system-map       : success!
common-resources: success!
log-shipping     : success!
logs            : success!
resources-details: success!
tracing-ingestion: success!
tracing-ingestion-edge: success!

Creating
Virtual Env
```

Pulling latest from GitHub

```
Deploying bulk #0, repositories: common-resources, log-shipping
.....
common-resources: success! (total 18.3 seconds)
log-shipping    : success! (total 69.7 seconds)

Deploying bulk #1, repositories: tracing-ingestion, resources-details, system-map, tracing-ingestion-edge, logs, alerting, java-tracer
.....
tracing-ingestion: success! (total 289.6 seconds)
resources-details: success! (total 220.3 seconds)
system-map       : success! (total 142.1 seconds)
tracing-ingestion-edge: success! (total 114.9 seconds)
logs            : success! (total 170.5 seconds)
alerting         : success! (total 184.7 seconds)
java-tracer     : success! (total 49.9 seconds)
```

Parallel deploy in bulks #1

Parallel deploy in bulks #2

```
Successfully deployed
overall time: 403.1198239326477
Shays-MacBook-Pro:sls_deploy shay$
```

# Test locally

- Unit tests
- Git pre-commit hooks
- *“Testing serverless from the trenches”*  
<https://tinyurl.com/testing-serverless>

Test

# Unit tests vs Integration tests

- Don't use service mocks

Test

# Testing Stack

# Test

- Linting    Flake8 ESLint
- Static type analysis   my[py]
- Unit testing   MOCHA  Jest

# Integration testing

API Integration test

End to End unit tests – using Cypress

Main problems:

- Deployment is slow
- Running the tests themselves is slow

Test

# Integration testing

- Each test in a dedicate AWS account – Isolation
- Parallel running of all tests (hundreds) – Speed
- No additional cost – creating the environment cost nothing, just the execution – Serverless power

*“SERVERLESS CONTINUOUS INTEGRATION IN THE ERA OF PARALLELISM”*  
<https://tinyurl.com/it-parallel>

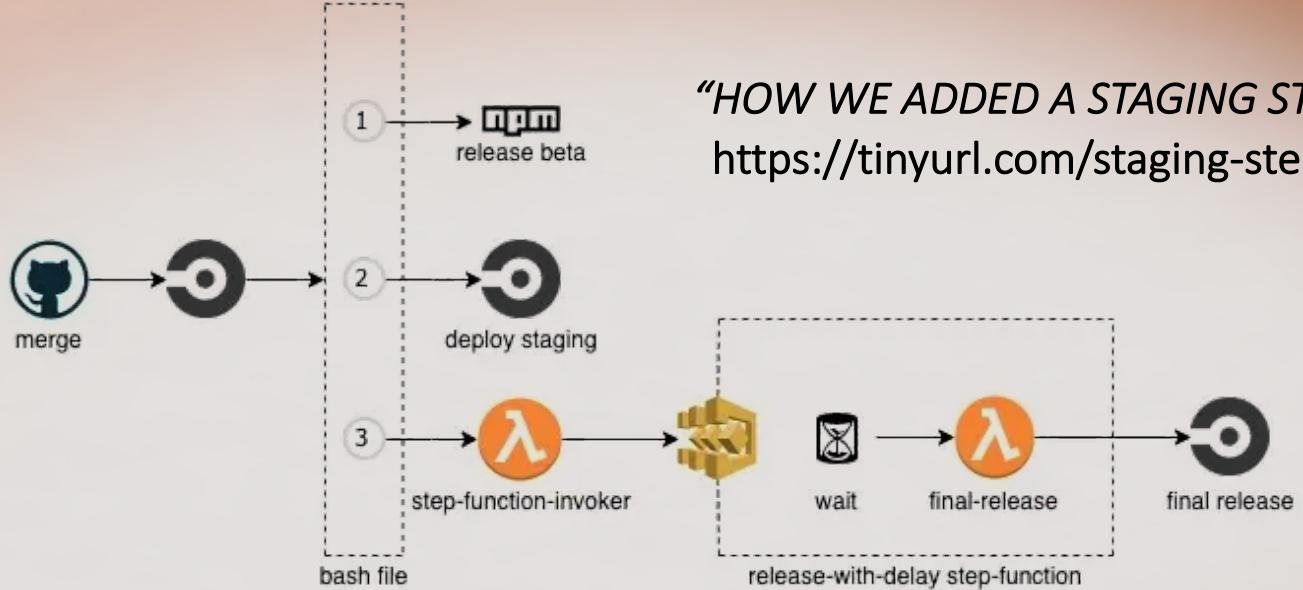
# Test

# Integration testing

- Using CF tags to avoid redeployment
- Still no good solution for initial deployments

Test

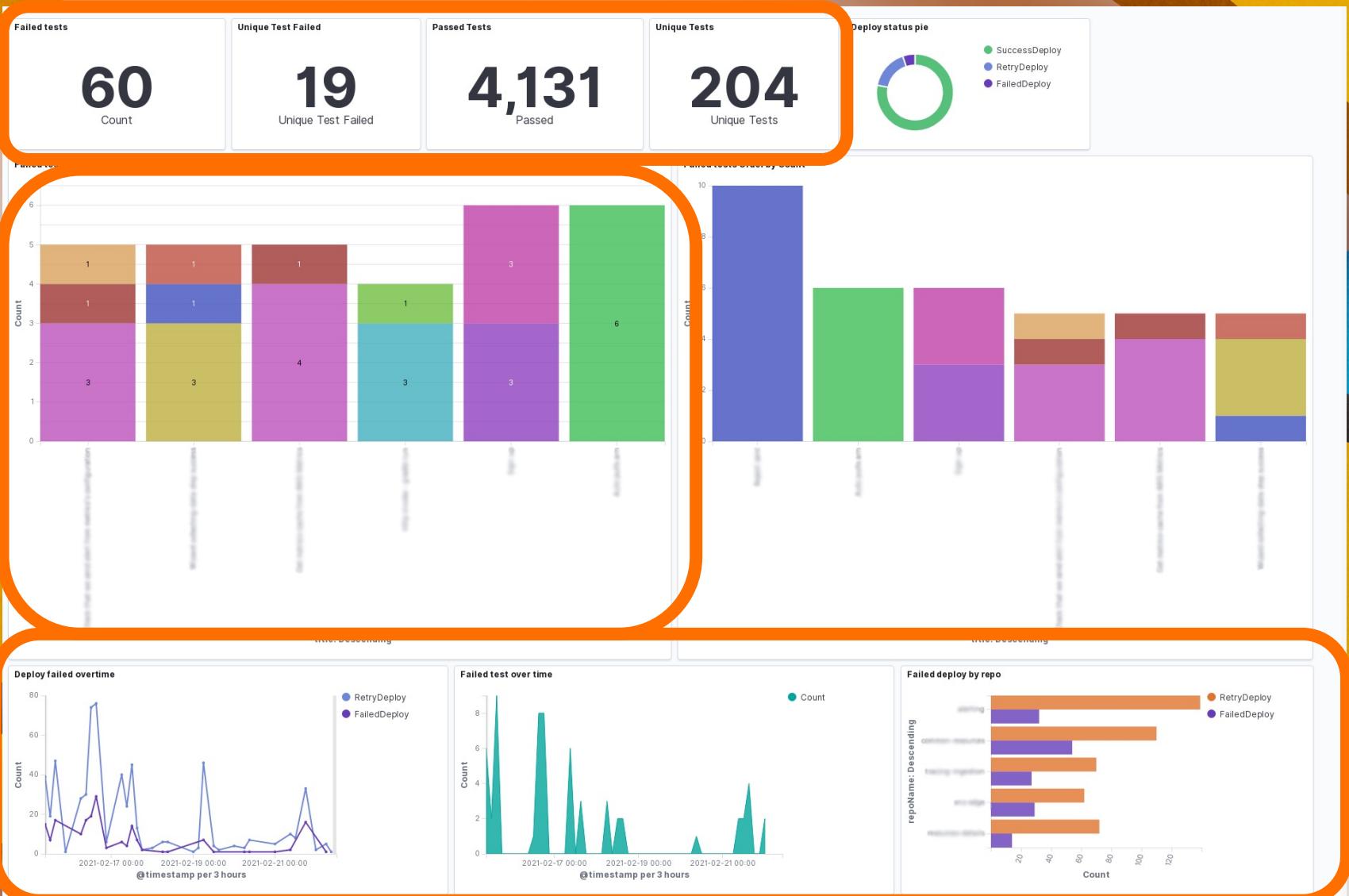
# Testing critical components



**"HOW WE ADDED A STAGING STEP"**  
<https://tinyurl.com/staging-step>

# Test

# Integration testing metrics



# Cleaning

- Really difficult!
- We use a combination of lumigo-cli and AWS-nuke

*"Reusing internal AWS accounts:  
Challenges and lessons learned"*  
<https://tinyurl.com/clean-aws>

# Test

# Release Gates

# Release

- Code Review
- Code Coverage
- Linting + Type testing
- Tests
- Merge to master  
→ deploy to prod



Changes approved

1 approving review by reviewers with write access. [Learn more.](#)

1 approval

uri-p approved these changes

All checks have passed

5 successful checks

[Hide all checks](#)

ci/circleci: e2e-test — Your tests passed on CircleCI! [Details](#)

ci/circleci: integration-test — Your tests passed on CircleCI! [Required Details](#)

ci/circleci: test — Your tests passed on CircleCI! [Required Details](#)

codecov/patch — 98.36% of diff hit (target 97.39%) [Details](#)

codecov/project — 97.58% (+0.18%) compared to e8bcfd2 [Details](#)

This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

[Resolve conflicts](#)

Conflicting files

requirements.txt

# Monitoring Serverless

# Monitor

# Serverless: “Monitoring in the Dark”





Use our own dog food

# Takeaways

- Use the power of Serverless (Parallel testing)
- Integration tests are the king
- Automate your deployment process so it's easy for developers

## Most used CLI commands by our users:

- Switch account
- Clear account
- Analyze cold starts

<https://github.com/lumigo-io/lumigo-CLI>

## lumigo-cli

A collection of helpful commands for working with AWS Lambda.

di oclif npm v0.14.0 downloads 283/week License Apache 2.0

- [Usage](#)
- [Commands](#)

### Usage

```
$ npm install -g lumigo-cli  
$ lumigo-cli COMMAND  
running command...  
$ lumigo-cli (-v|--version|version)  
lumigo-cli/0.14.0 darwin-x64 node-v10.16.0  
$ lumigo-cli --help [COMMAND]  
USAGE  
  $ lumigo-cli COMMAND  
...
```

### Commands

- [lumigo-cli analyze-lambda-cost](#)
- [lumigo-cli help \[COMMAND\]](#)
- [lumigo-cli list-kinesis-shards](#)
- [lumigo-cli list-lambda](#)
- [lumigo-cli powertune-lambda](#)
- [lumigo-cli replay-sqs-dlq](#)
- [lumigo-cli sls-remove](#)

# THANK YOU !

## Questions?

---

Efi Merdler-Kravitz  
VP of Engineering

✉️ [efi@lumigo.io](mailto:efi@lumigo.io)  
🐦 [@TServerless](https://twitter.com/TServerless)



imgflip.com