AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

DOP310

Enabling decentralized development teams with a shared services platform

Andrew Baird (he/him)
Principal Solutions Architect
Amazon Web Services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Problem statement
- AWS services that can help
- Common patterns
- Crawl, walk, run options
- Conclusion

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Problem statement

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Problem statement

Development speed needs to be balanced against the requirements of company policies (e.g., security, compliance, operations) . . .

. . . or does it?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Problem statement

Company policies (if valid) should be **nonnegotiable**

Development speed **should slow** if meeting policies becomes **difficult**

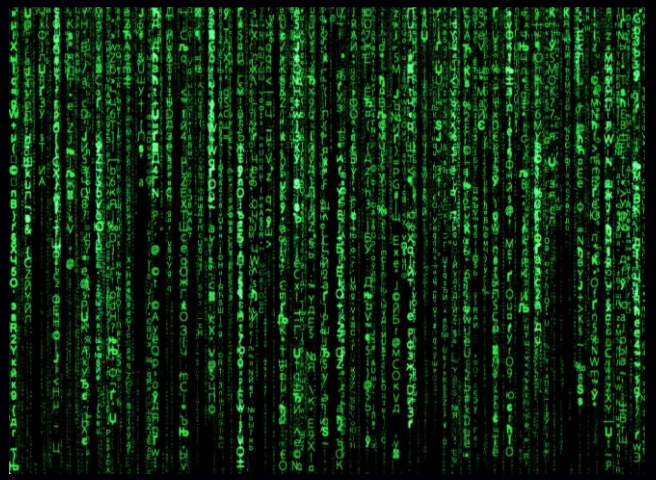
So, focus on offering development teams **abstractions**
that make following company policies **easier** and **faster**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example abstractions that support DevOps



Frameworks



Command Line Interfaces



Deployment services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building your own CLI, framework, or deployment service from the beginning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

. . . just kidding

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

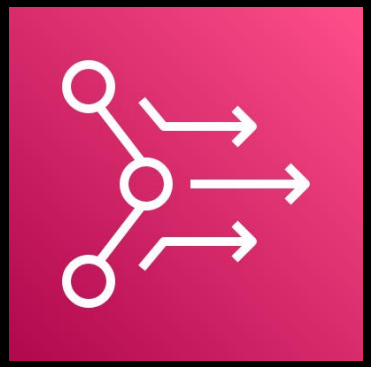
Existing AWS abstractions to the rescue!



AWS SAM
AWS CDK
AWS CloudFormation



AWS Copilot
AWS SAM CLI



AWS Proton
AWS CodeDeploy

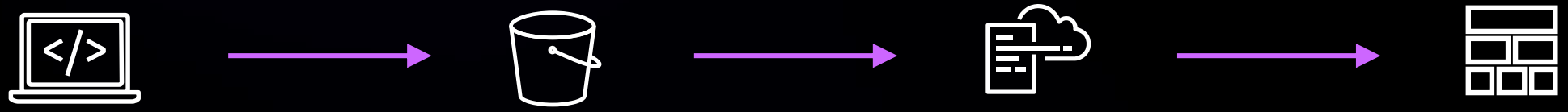
Frameworks | **Command line interfaces** | **Deployment services**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS CloudFormation

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview: AWS CloudFormation



Code in YAML or JSON directly or use sample templates

Upload local files or files from an Amazon S3 bucket

Create stack using console, API, or CLI

Stacks and resources are provisioned

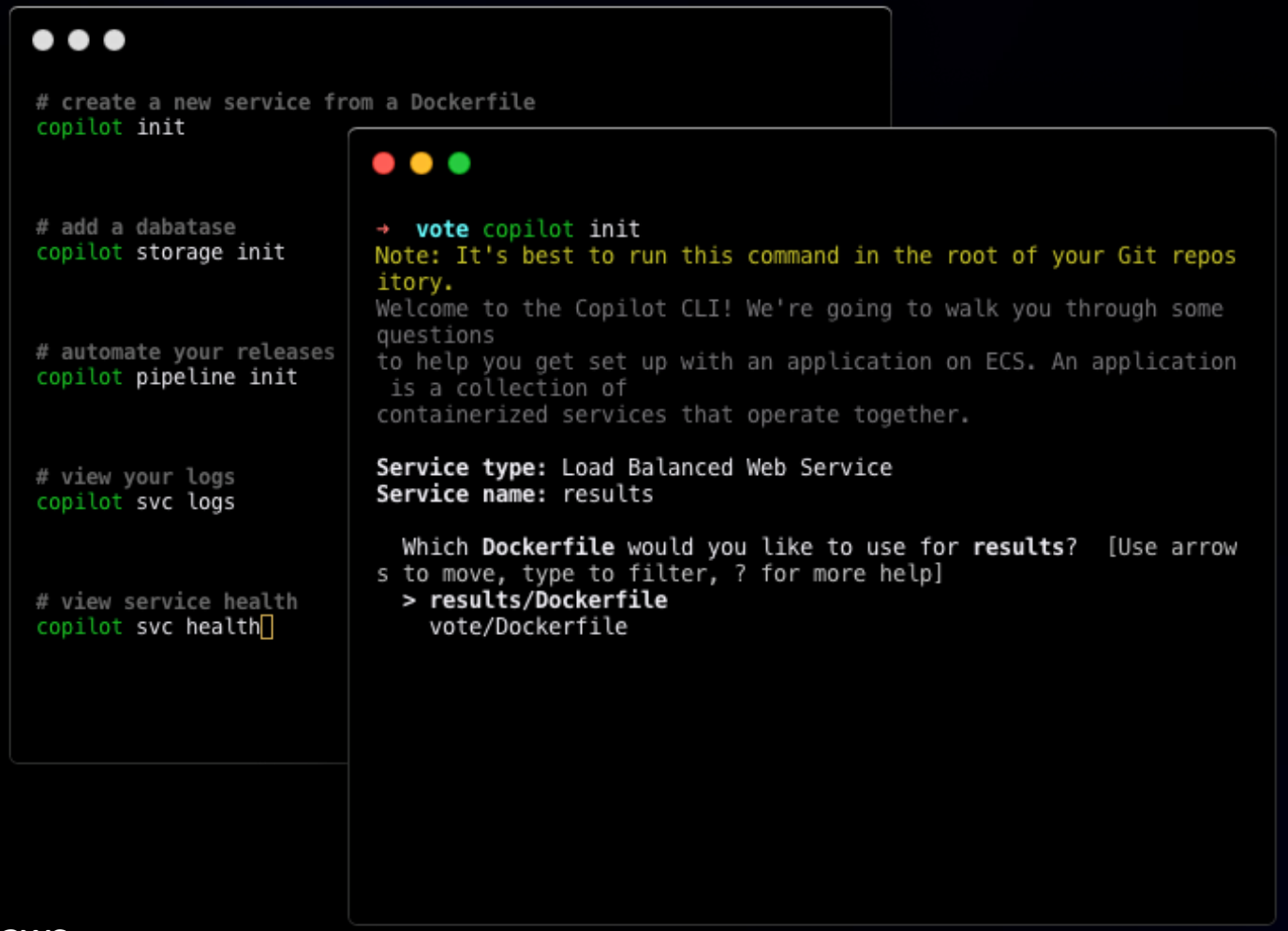
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Copilot

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview: AWS Copilot CLI

LAUNCH AND MANAGE CONTAINERS ON AWS FARGATE WITH AMAZON ECS



Think in terms of **architecture**, not infrastructure

Develop with simple, powerful configurations

Release with a continuous delivery pipeline

Operate with auto scaling, alarms, metrics, and logs

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Copilot walkthrough

```
Admin:~/environment/aws-copilot-sample-service (master) $ copilot init --app reinvent-demo
Welcome to the Copilot CLI! We're going to walk you through some questions
to help you get set up with a containerized application on AWS. An application is a collection of
containerized services that operate together.

Workload type: Load Balanced Web Service
Service name: api
Dockerfile: ./Dockerfile
Ok great, we'll set up a Load Balanced Web Service named api in application reinvent-demo listening on port 80.

✔ Created the infrastructure to manage services and jobs under application reinvent-demo..

✔ The directory copilot will hold service manifests for application reinvent-demo.

✔ Wrote the manifest for service api at copilot/api/manifest.yml
Your manifest contains configurations like your container size and port (:80).

✔ Created ECR repositories for service api..

All right, you're all set for local development.

  Would you like to deploy a test environment? [? for help] (y/N)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Copilot walkthrough

```
Deploy: Yes

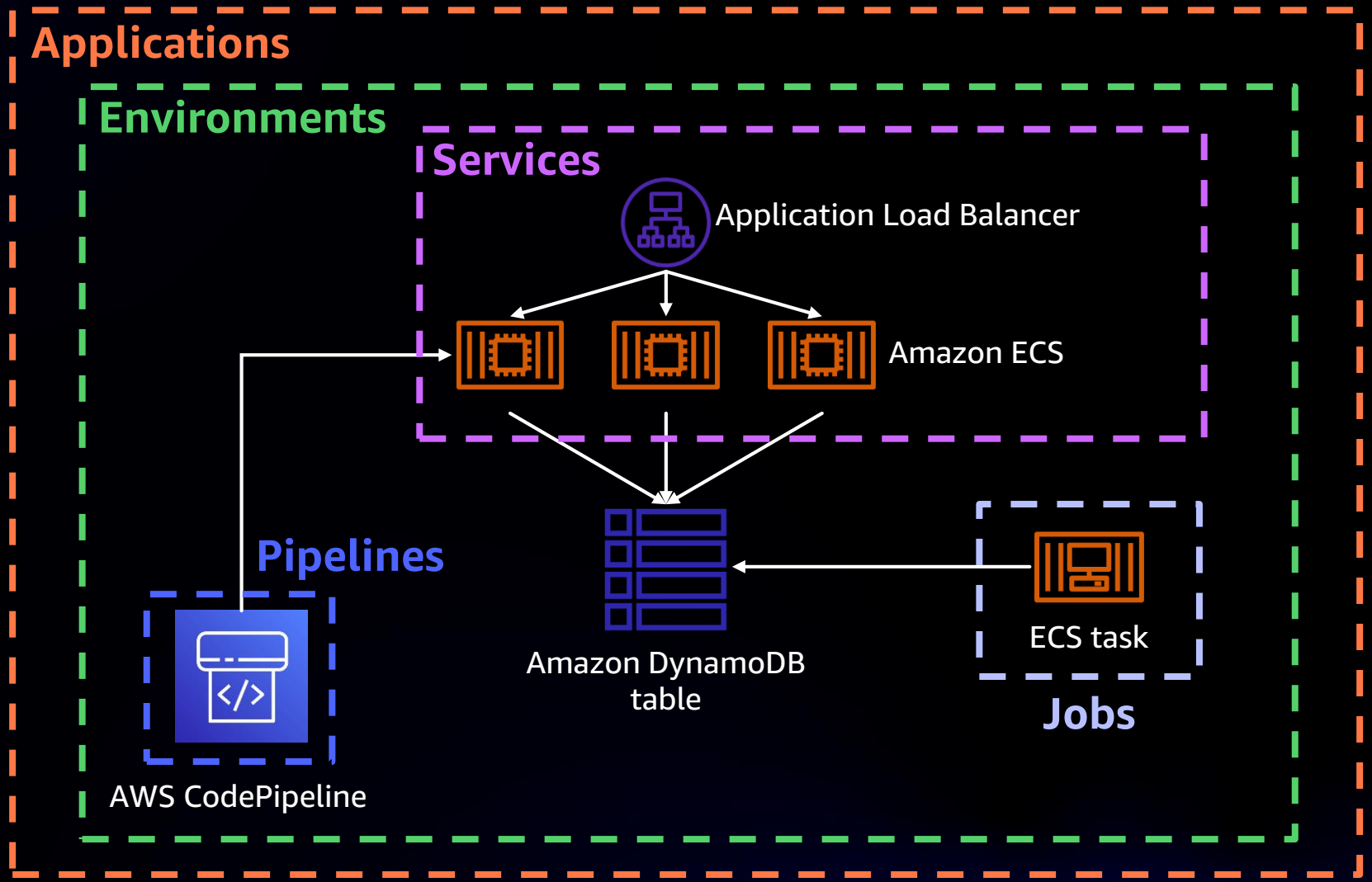
✔ Linked account [redacted] and region us-east-1 to application reinvent-demo..

✔ Proposing infrastructure changes for the reinvent-demo-test environment.
- Creating the infrastructure for the reinvent-demo-test environment.    [create complete]  [81.2s]
  - An IAM Role for AWS CloudFormation to manage resources               [create complete]  [15.6s]
  - An ECS cluster to group your services                                [create complete]  [8.7s]
  - Enable long ARN formats for the authenticated AWS principal          [create complete]  [6.1s]
  - An IAM Role to describe resources in your environment                [create complete]  [15.9s]
  - A security group to allow your containers to talk to each other      [create complete]  [8.7s]
  - An Internet Gateway to connect to the public internet                [create complete]  [15.3s]
  - Private subnet 1 for resources with no internet access               [create complete]  [15.4s]
  - Private subnet 2 for resources with no internet access               [create complete]  [15.4s]
  - Public subnet 1 for resources that can access the internet           [create complete]  [15.4s]
  - Public subnet 2 for resources that can access the internet           [create complete]  [15.4s]
  - A Virtual Private Cloud to control networking of your AWS resources  [create complete]  [17.8s]
✔ Created environment test in region us-east-1 under application reinvent-demo.
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Working with AWS Copilot

KEY CONCEPTS



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Proton

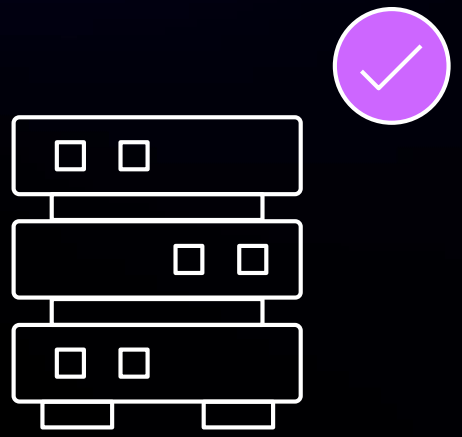
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview AWS Proton

- Created with platform teams in mind
- Automates provisioning and deployment
- Standardizes infrastructure
- Integrates with CI/CD

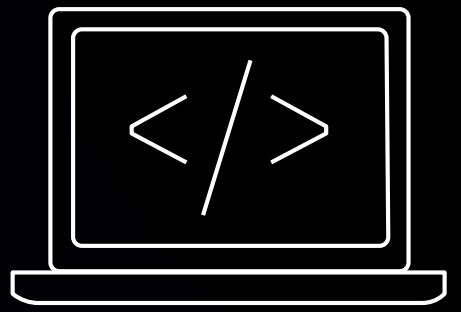
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AW Proton user personas



Administrators

Build and publish templates that model infrastructure, CI/CD pipelines, and observability tools

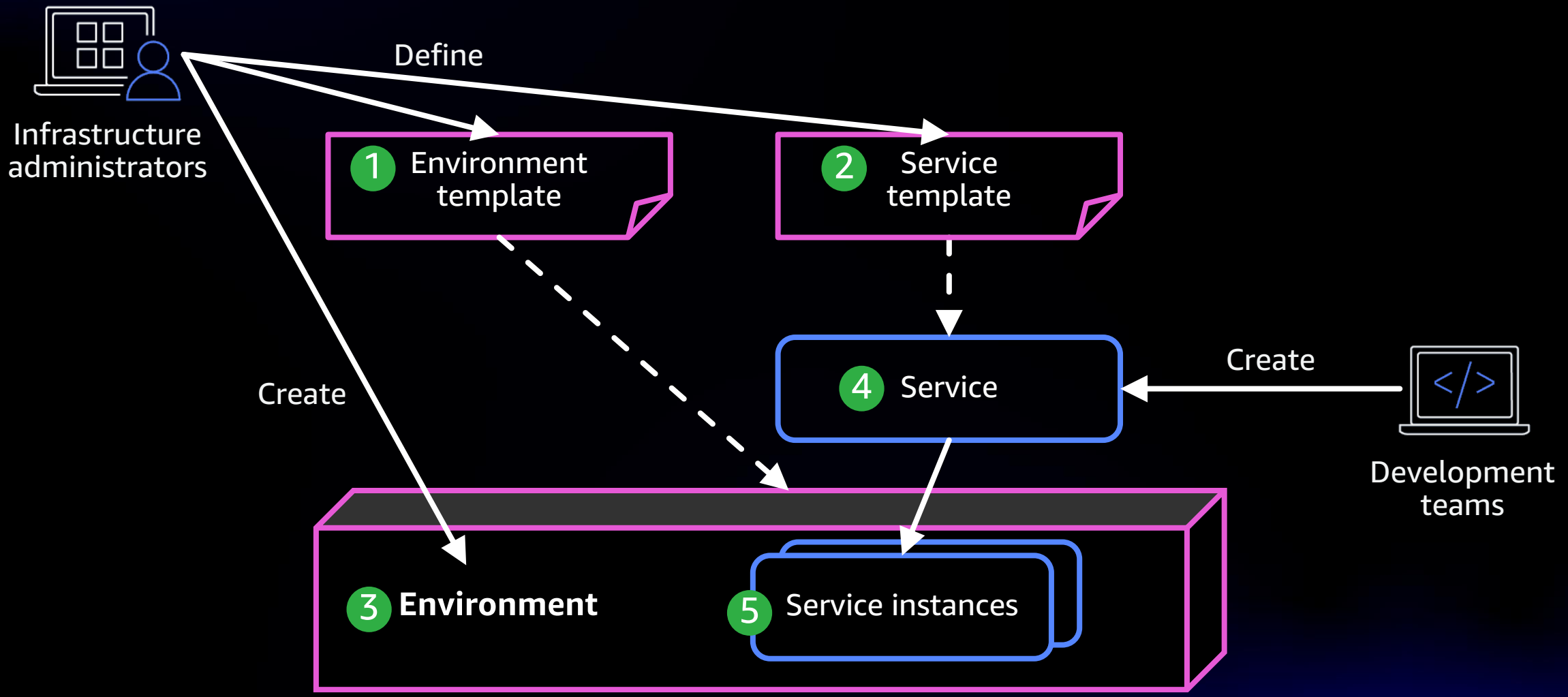


Developers

Find and deploy complete services for your applications

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Working with AWS Proton resources



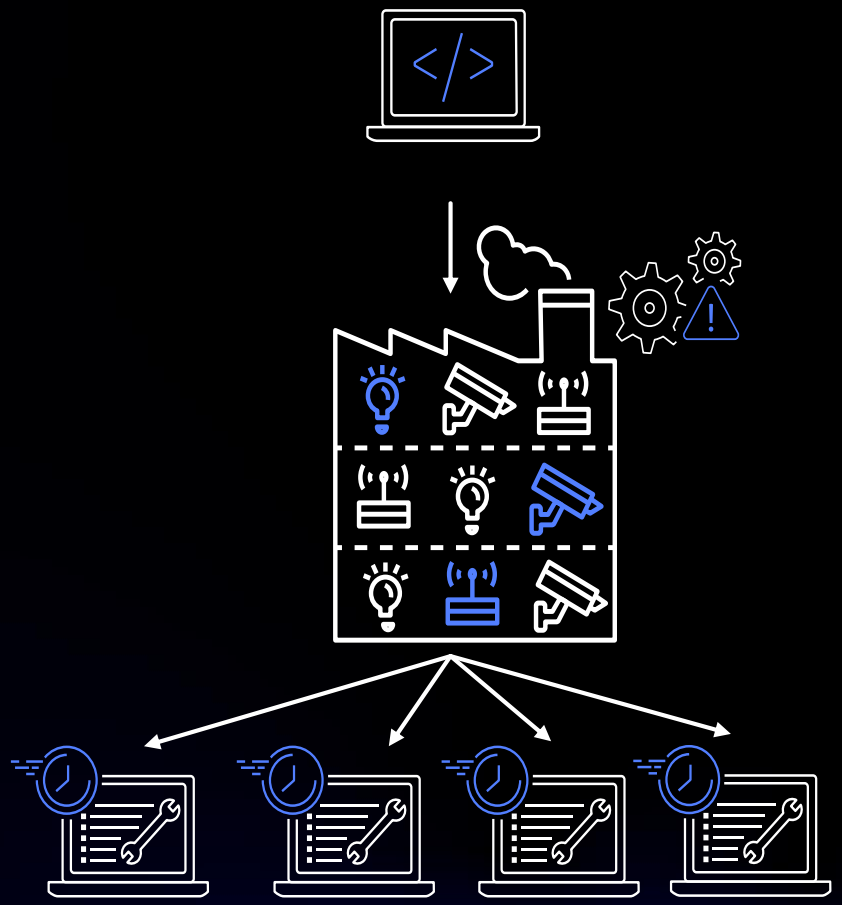
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Platform ownership patterns to consider

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Instrument AWS accounts as “the platform”

- AWS accounts owned and operated by application teams
- Additional instrumentation provided in those accounts by platform team
  - Supporting infrastructure (e.g., networking, security, DNS, company standards, etc.)
  - AWS Proton environments/applications
  - AWS Service Catalog products
- Consider: AWS Control Tower, AWS Organizations, CloudFormation StackSets, AWS Config conformance packs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managed container clusters as “the platform”

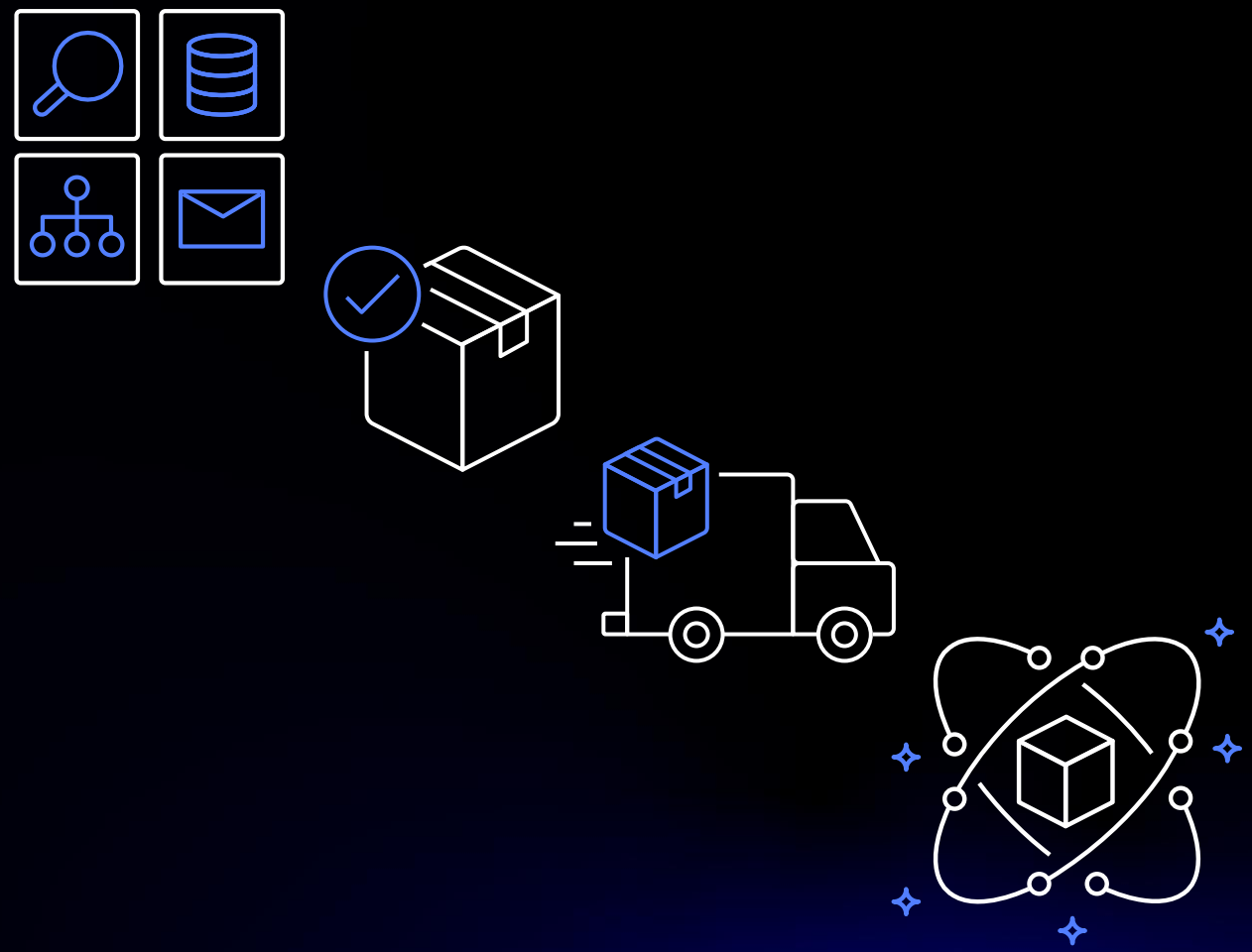
- AWS Accounts owned and operated by platform team
- Application teams deploy applications onto multi-tenant clusters managed by platform team
  - CI/CD
  - Ingress control
  - Access management
  - Operations visibility
- Deployment mechanisms for surrounding infrastructure
  - AWS CloudFormation/AWS CDK with CI/CD
  - AWS Proton environments/applications



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deployable application patterns as “the platform”

- Your choice for which team owns the AWS accounts where applications run
  - Shared accounts = platform team owns
  - Many accounts = application teams own
- Platform team creates ready-made applications and deployment mechanisms that application teams customize and consume
  - AWS Proton
  - AWS CloudFormation/AWS Service Catalog
  - AWS CDK constructs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawl, walk, run

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawl – Demonstrate value

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawl – Find one team, improve their life, and repeat

YOUR FIRST STEPS

1. Find a typical use case
2. Define the interaction model
   - Infrastructure as code/CLI
   - Platform infrastructure
   - Deployment
3. Agree on ownership outcomes (they must scale when adoption grows)



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawl – Choosing the right service

YOUR FIRST STEPS

AWS CloudFormation

- Generic, flexible
- Double-edged ambiguity
- Good choice when wanting to own “nothing” or “everything”



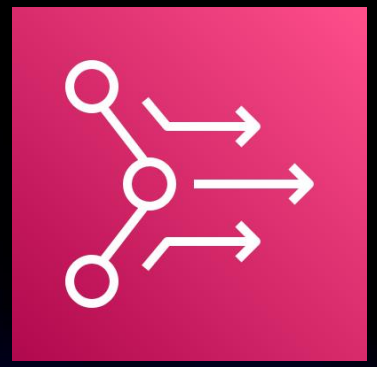
AWS Copilot

- For container-based applications
- Development teams do not want to own or care about infrastructure



AWS Proton

- Good when you want application and infrastructure ownership to be kept separate
- Application teams need or want opinionated patterns



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawl – Helpful resources

STARTING POINTS

AWS CloudFormation

github.com/awslabs/aws-solutions-constructs

1. Review prebuilt collection of architecture patterns
2. Pluggable, extensible
3. Maximum flexibility, thinly abstracted

AWS Copilot

github.com/aws-samples/aws-copilot-sample-service

1. Understand the **AWS Copilot-created** AWS infrastructure
2. Surround AWS Copilot-created applications with **company-relevant infrastructure**

AWS Proton

github.com/aws-samples/aws-proton-sample-templates

1. Be specific about what is in an **environment** and what is in an **application**
2. Validate environment choices with second team onboarded

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Walk – Scaling up

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Walk – Document, market, scale

YOUR NEXT STEPS

1. **Document** the platform you've built
   - Onboarding steps (self-service, automated)
   - User guide/API documentation
   - Contact mechanisms
   - Publish the code (innersource, or open source?)
2. **Market** the platform internally
   - Use success to earn sponsor from leaders (quantify value, risk aversion, efficiency)
   - Go on a "roadshow" to recruit teams **and take feedback**
3. Prove your approach can **scale**
   - How many adopters is too many? (hint: no limit)
   - Contemplate supporting hundreds of teams, across various scenarios (e.g., application outages, AWS service events, migrations, zero-day patching)
   - Where are the bottlenecks? (onboarding, learning curve, deployment, support/on call, etc.)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Run – Flywheels

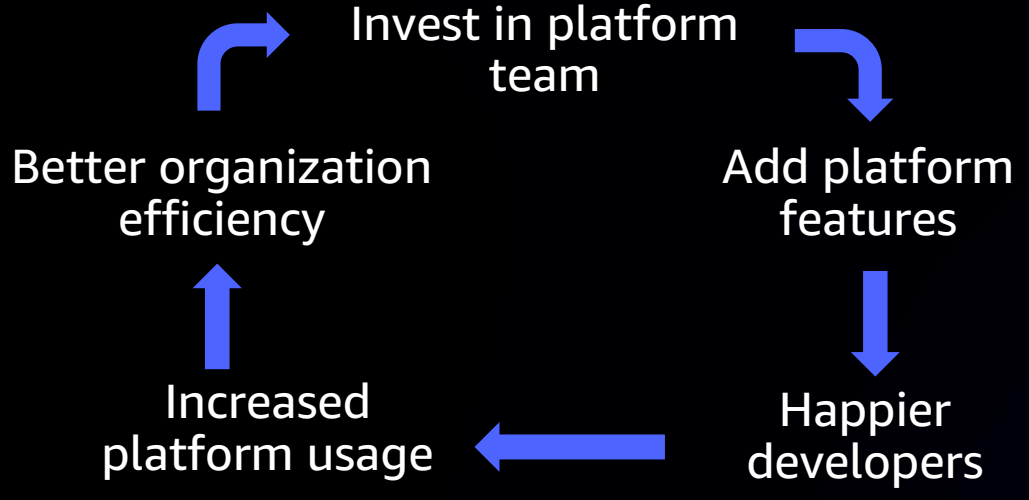
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

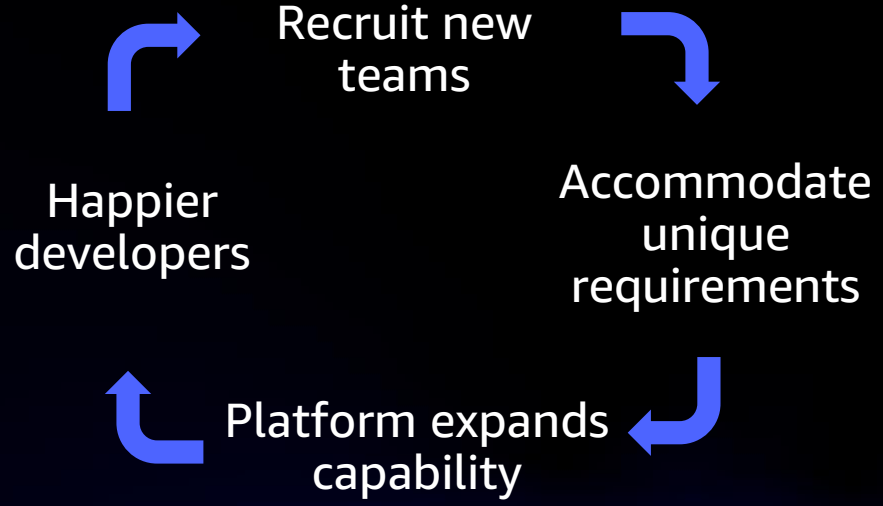
Run – Execute your flywheel

CAPTURE MOMENTUM FOR LONG-TERM SUCCESS

Flywheel: A virtuous cycle that generates its own increase in momentum

Example flywheels for platform teams





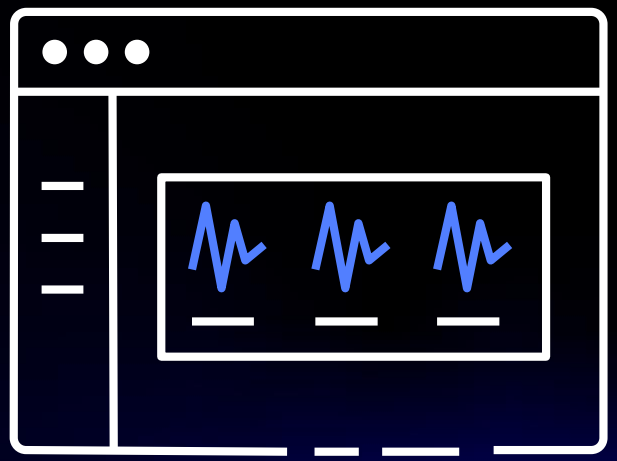
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Summary

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Summary

1. Abstractions succeed when they make hard work easier
2. Create abstractions that
   - Provide **adherence to policies** by default
   - Deploy **patterned architectures** using preferred **interaction model**
   - Ensure **ownership is aligned** to organization structure
3. Treat the abstractions like a **product**
   - Document
   - Market
   - Support



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.