AWS Copilot

Primož Vampelj

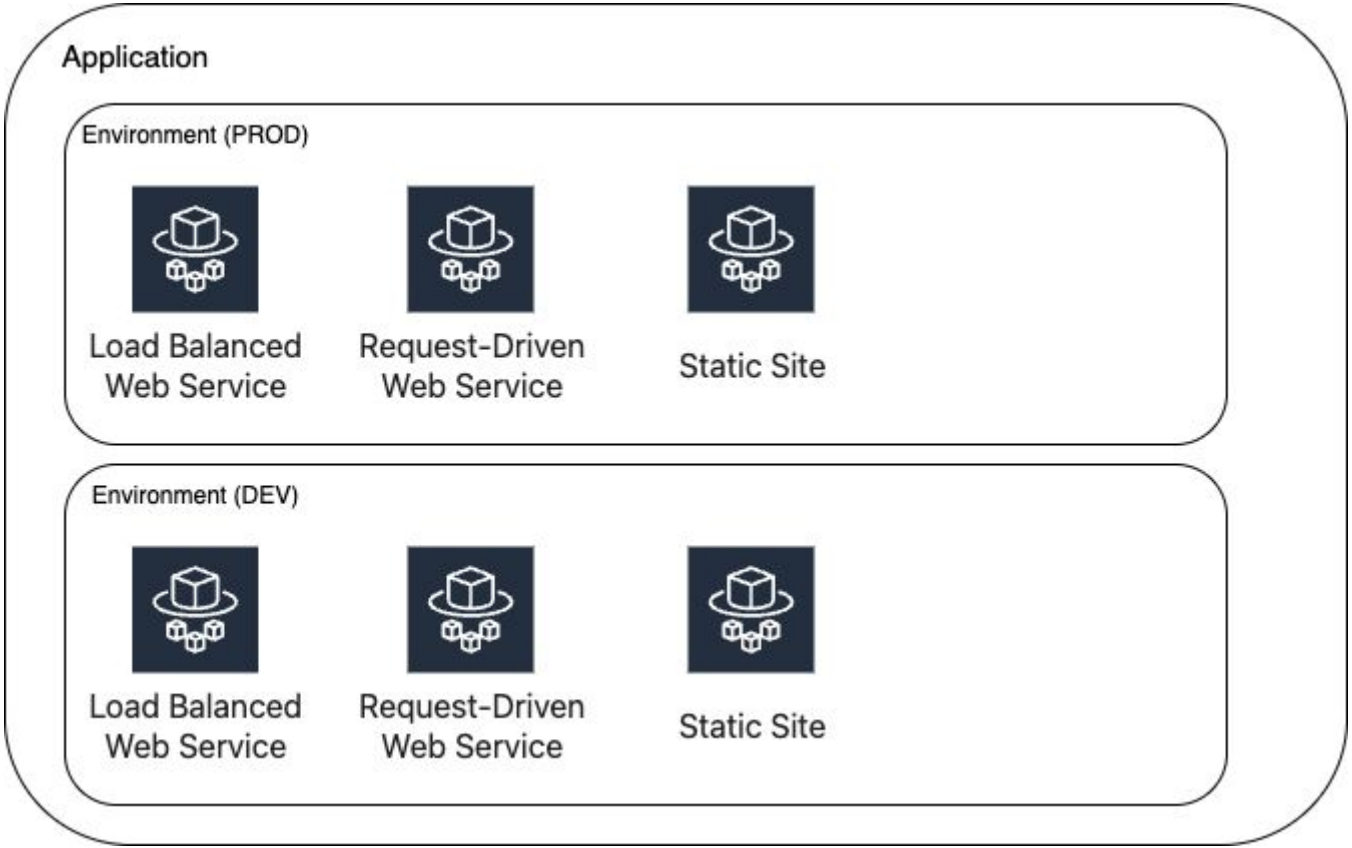
What is AWS Copilot?

An AWS CLI tool that enables developers to deploy and manage simple applications backed by AWS Fargate or App Runner to AWS with supporting infrastructure

AWS Copilot CLI

```
[~] copilot --help
🧑‍✈️ Launch and manage applications on Amazon ECS and AWS Fargate.

Commands
  Getting Started 🌱
    init        Create a new ECS application.
    docs        Open the copilot docs.

  Develop ✨
    app         Commands for applications.
                Applications are a collection of services and environments.

    env         Commands for environments.
                Environments are deployment stages shared between services.

    svc         Commands for services.
                Services are long-running Amazon ECS services.

  Release 🚀
    pipeline    Commands for pipelines.
```

High level overview



Application

- Groups connected services, environments and pipelines
- How to start?

```
$ copilot app init                                  \
  --domain my-awesome-app.aws                       \
  --resource-tags department=MyDept,team=MyTeam     \
  --permissions-boundary my-pb-policy
```

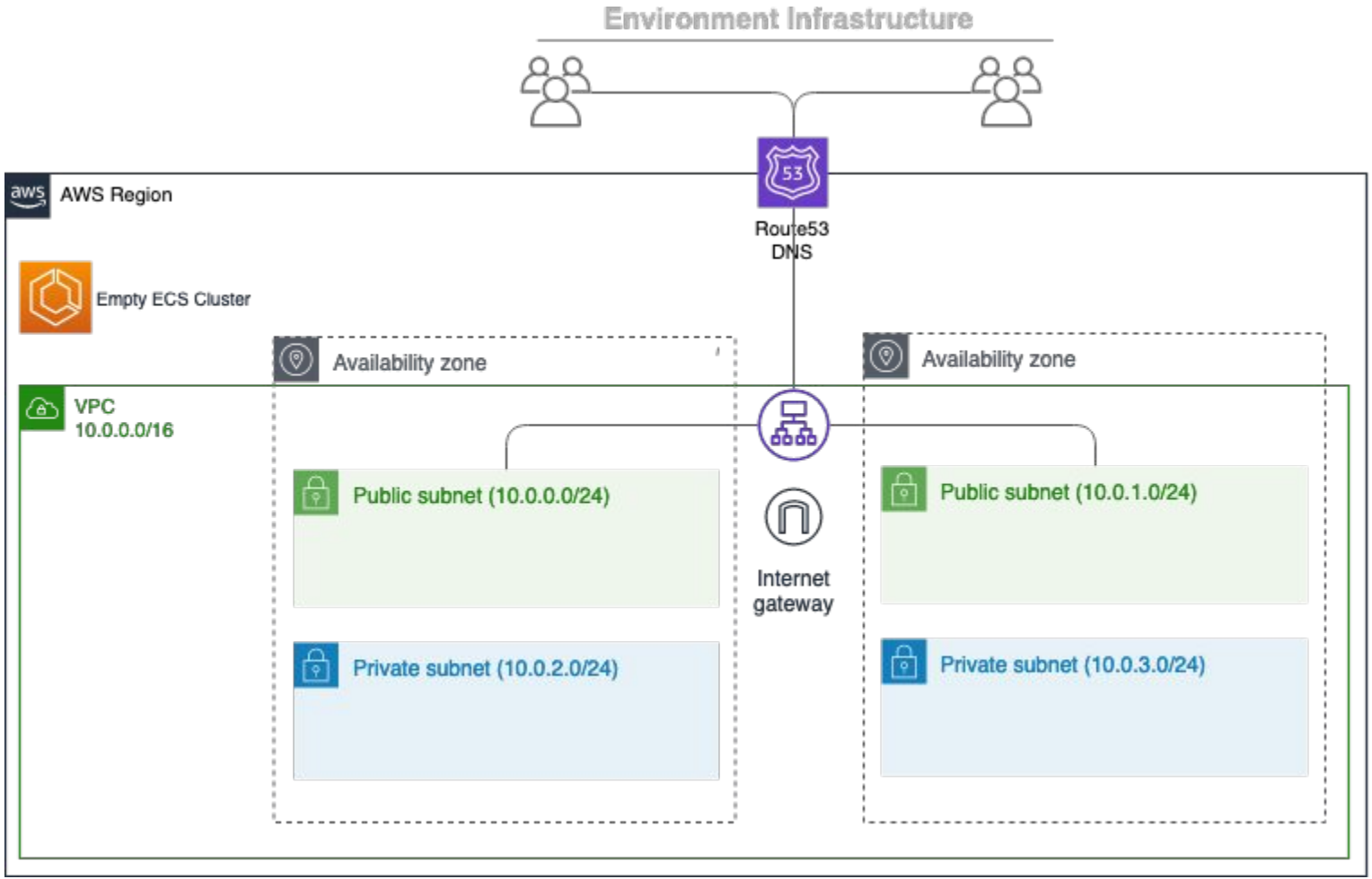
What gets created

```
$ copilot app show
About

  Name              vote
  Version           v1.1.0
  URI               vote-app.aws

Environments

  Name              AccountID           Region
  ----              ---------           ------
  test              000000000000        us-east-1

Workloads

  Name              Type                        Environments
  ----              ----                        ------------
  collector         Load Balanced Web Service   prod
  aggregator        Backend Service             test, prod

Pipelines

  Name
  ----
```

Environments

- You can create it from scratch or import your existing infrastructure
- Define your:
  - Networking (VPC, Subnets)
  - Load Balancer SG + certificates
- Environments have their own manifest file and can be re-deployed

```
❯ copilot env init
Environment name: test
Credential source: [profile vampelj_admin]

  Would you like to use the default configuration for a new environment?
    - A new VPC with 2 AZs, 2 public subnets and 2 private subnets
    - A new ECS Cluster
    - New IAM Roles to manage services and jobs in your environment
  [Use arrows to move, type to filter]
    Yes, use default.
    Yes, but I'd like configure the default resources (CIDR ranges, AZs).
  > No, I'd like to import existing resources (VPC, subnets).
```


Environment Infrastructure
AWS Region
Route53
DNS
Empty ECS Cluster
Availability zone
Availability zone
VPC
10.0.0.0/16
Public subnet (10.0.0.0/24)
Public subnet (10.0.1.0/24)
Internet
gateway
Private subnet (10.0.2.0/24)
Private subnet (10.0.3.0/24)

Services

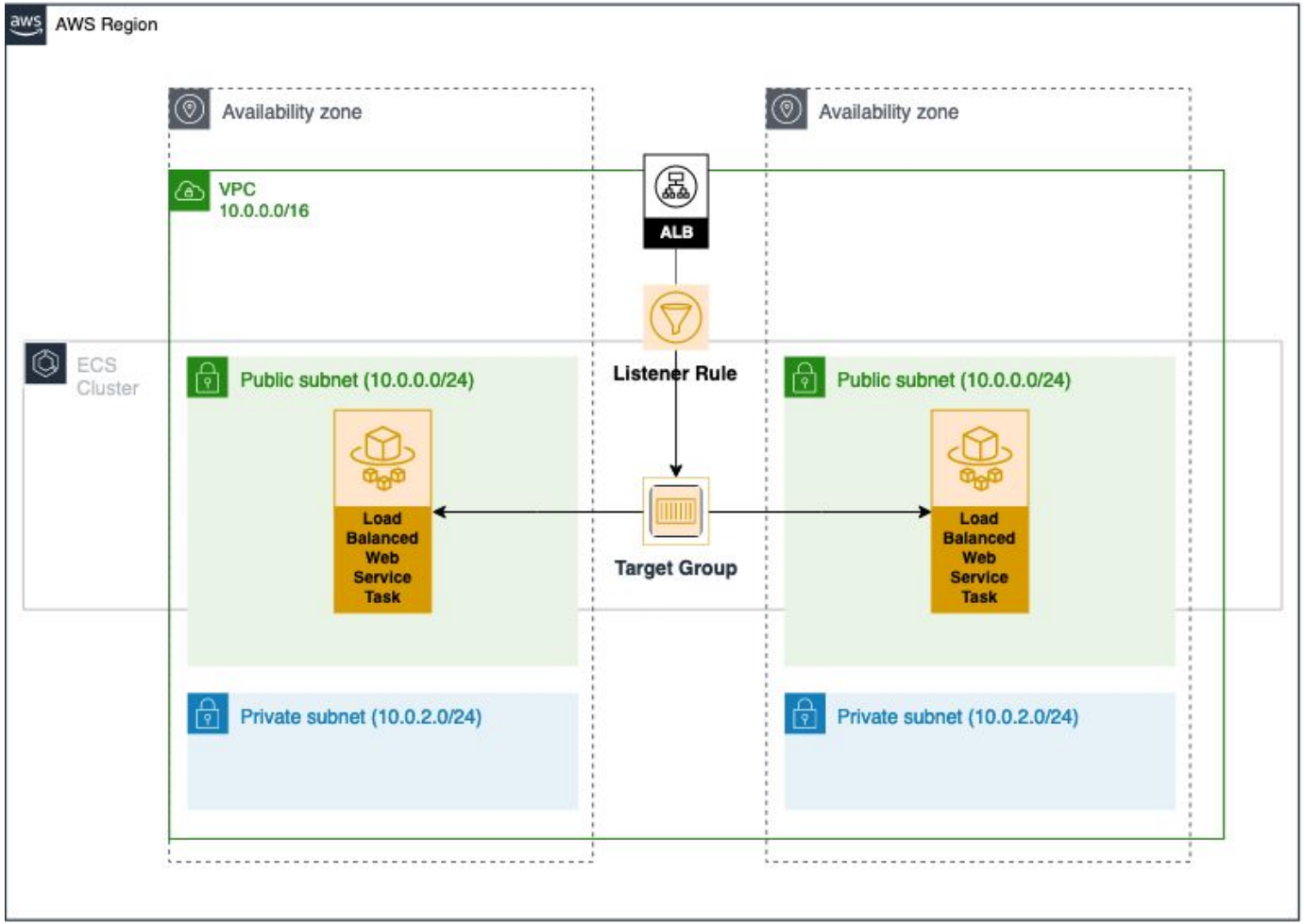
Internet facing:

- Request-Driven Web Service (App Runner)
- Static Site (S3 + CloudFront)
- **Load Balanced Web Service** (ALB and/or NLB, ECS Fargate)

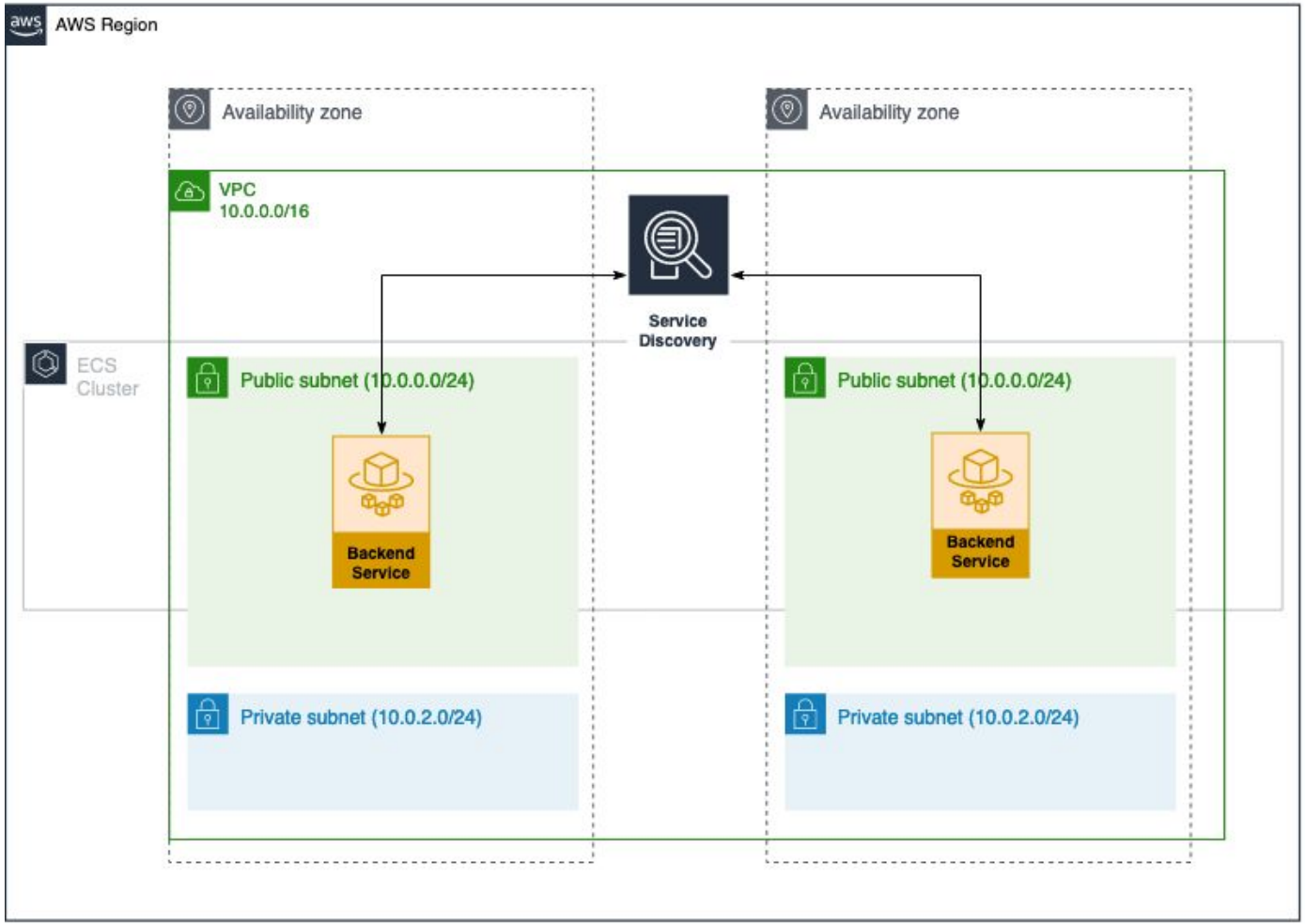
Private:

- Backend Service (ECS Fargate)
- Worker Service (SQS, ECS Fargate)

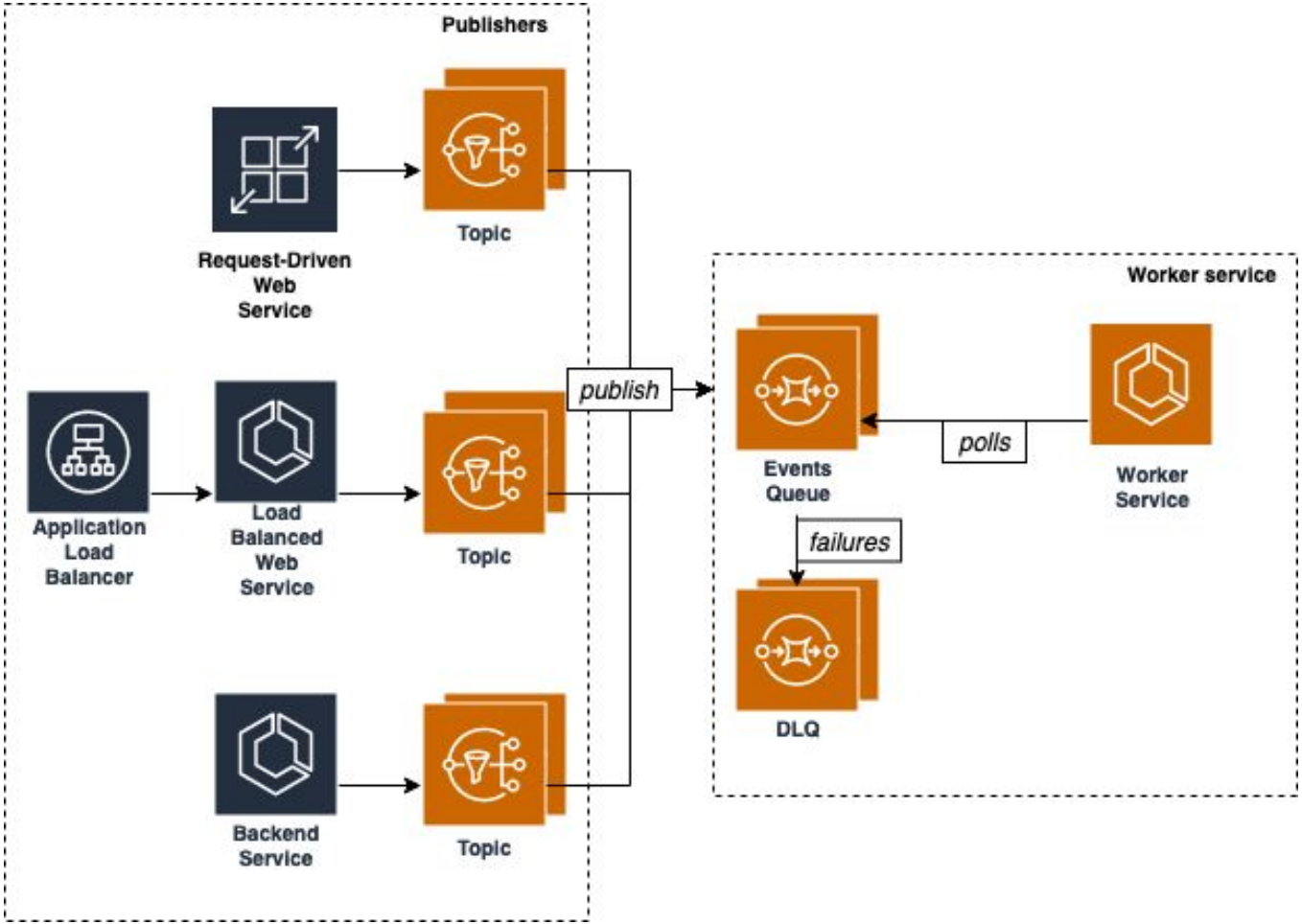
Load Balanced Web Service Infrastructure



Backend Service Infrastructure



Worker service



```yaml
name: front-end
type: Load Balanced Web Service

image:
  # Path to your service's Dockerfile.
  build: ./Dockerfile
  # Port exposed through your container to route traffic to it.
  port: 8080

http:
  # Requests to this path will be forwarded to your service.
  # To match all requests you can use the "/" path.
  path: '/'
  # You can specify a custom health check path. The default is "/"
  # healthcheck: '/'

# Number of CPU units for the task.
cpu: 256
# Amount of memory in MiB used by the task.
memory: 512
# Number of tasks that should be running in your service.
count: 1
```

```yaml
# Optional fields for more advanced use-cases.
#
variables:                    # Pass environment variables as key value pairs.
  LOG_LEVEL: info

#secrets:                         # Pass secrets from AWS Systems Manager (SSM) Pa
#  GITHUB_TOKEN: GH_SECRET_TOKEN  # The key is the name of the environment variabl
                                  # the value is the name of the SSM parameter.

# You can override any of the values defined above by environment.
environments:
  prod:
    count: 2               # Number of tasks to run for the "prod" environment.
```

Pipelines

- AWS Code deploy pipeline
- Source => Build => Deploy (to multiple environments)

```
$ copilot pipeline init
$ git add copilot/ && git commit -m "Adding pipeline artifacts" && git push
$ copilot pipeline deploy
```

```yaml
name: test-test-pipeline
version: 1

# This section defines your source, changes to which trigger your pipeline.
source:
  provider: Bitbucket
  properties:
    branch: develop
    repository: https://bitbucket.org/vampelj/test-test
stages:
  - name: dev
    pre_deployments:
      db_migration:
        buildspec: copilot/pipelines/test-test/db_migrations.yml
    post_deployments:
      celery_beat:
        buildspec: copilot/pipelines/test-test/celery_beat.yml
  - name: prod
    requires_approval: true
    pre_deployments:
      db_migration:
        buildspec: copilot/pipelines/test-test/db_migrations.yml
    post_deployments:
      celery_beat:
        buildspec: copilot/pipelines/test-test/celery_beat.yml
```

Extras

- Storage: S3, EFS, DynamoDB, Aurora
- Secrets: SSM Parameter store / Secret Manager
- Support for sidecars (logging, Grafana Alloy)
- Overrides (YAML Patch Overrides / CDK Overrides)
- Jobs (scheduled)

The good, the bad and the ugly

+ Easy to use and fast to deploy
+ Empowers developers
+ Great for simpler apps / infrastructure

o Can become cumbersome to manage a lot of manifest files
o Cloudformation takes its time to deploy

- Failed deploys can be a pain to fix

Tips

- Watch out for version mismatch
- READ THE DOCUMENTATION !!!
- If it seems wrong, it probably is!

Want to learn more

https://aws.github.io/copilot-cli/

Questions?

Thank you !