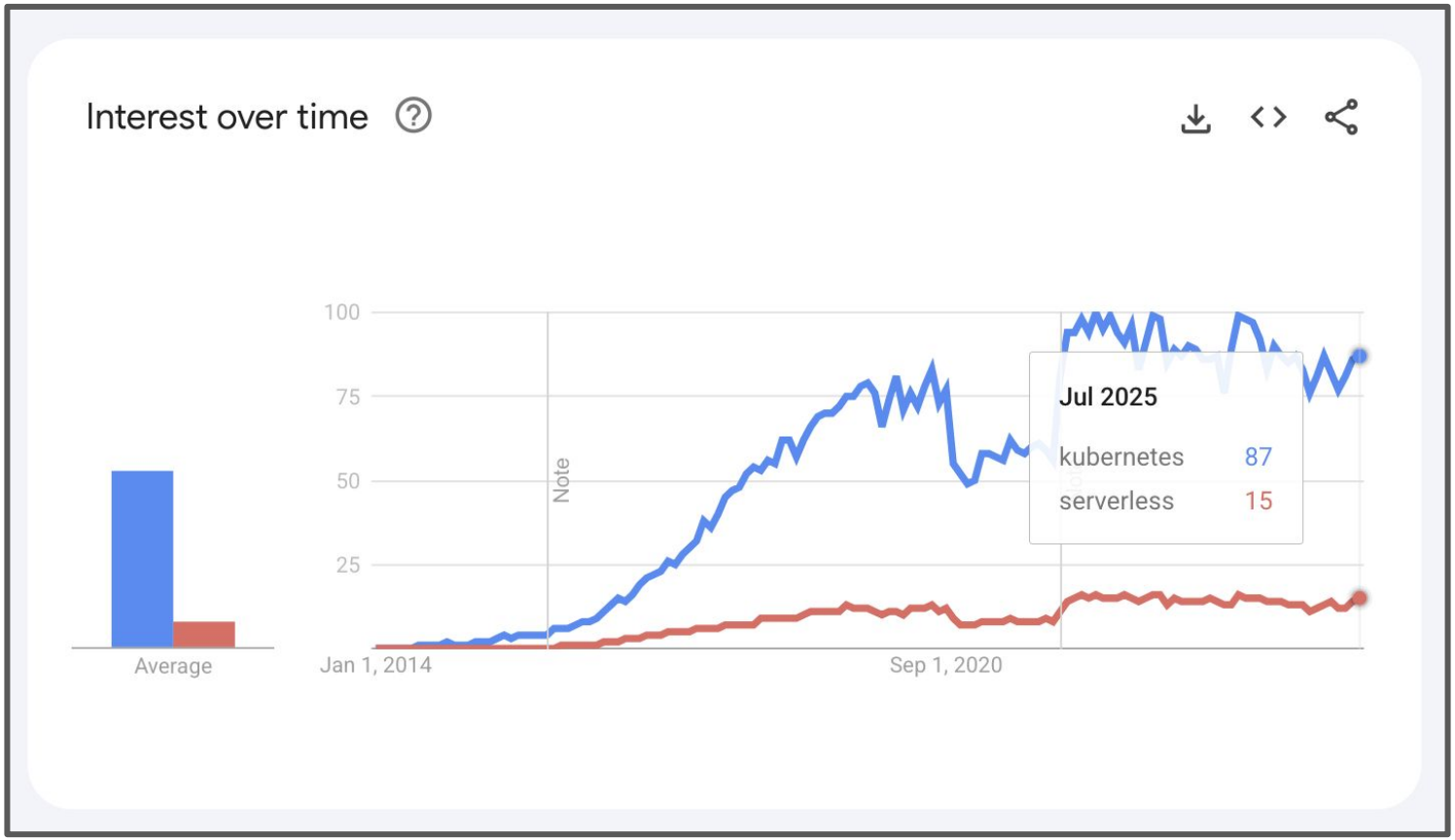
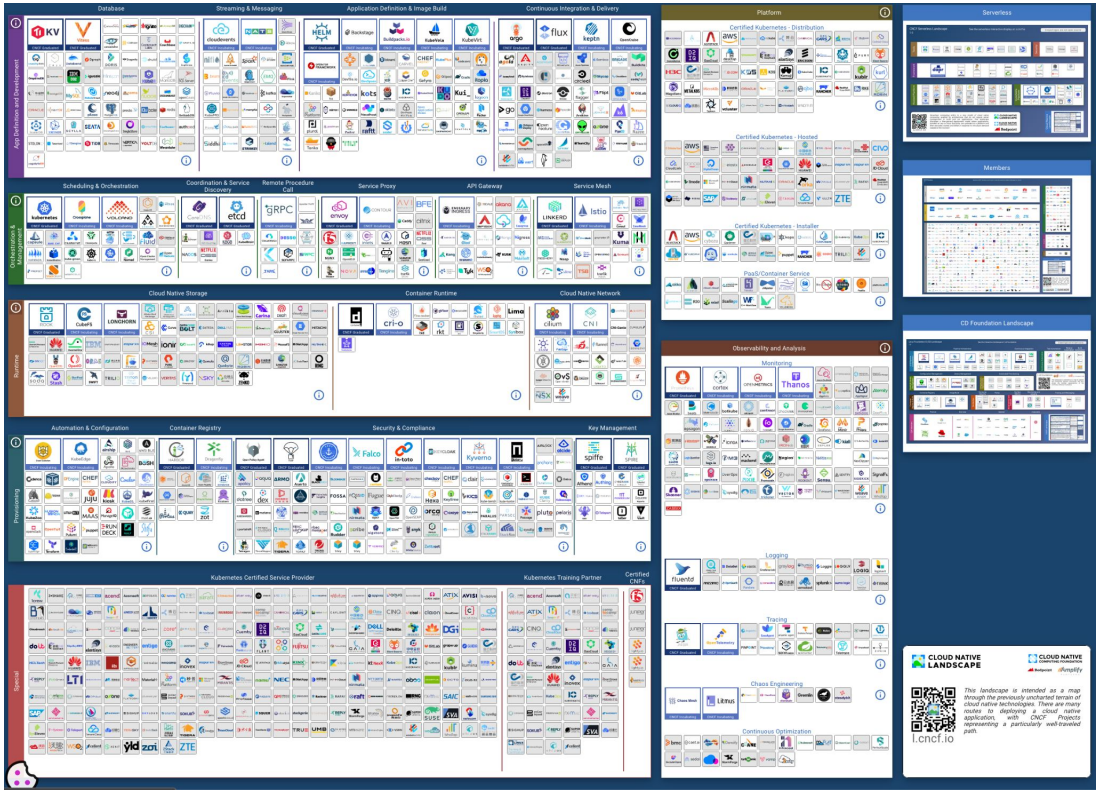
Applying a serverless mindset to internal developer platforms

80%



@erlendekern

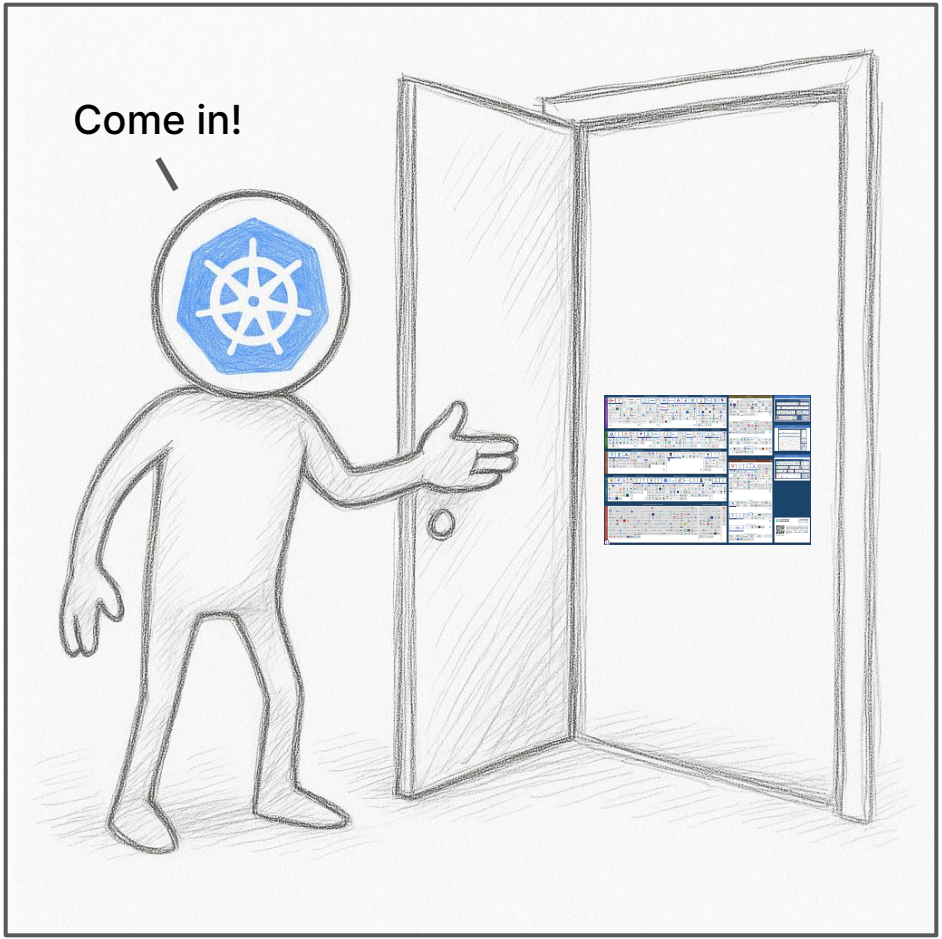
Capra



https://landscape.cncf.io

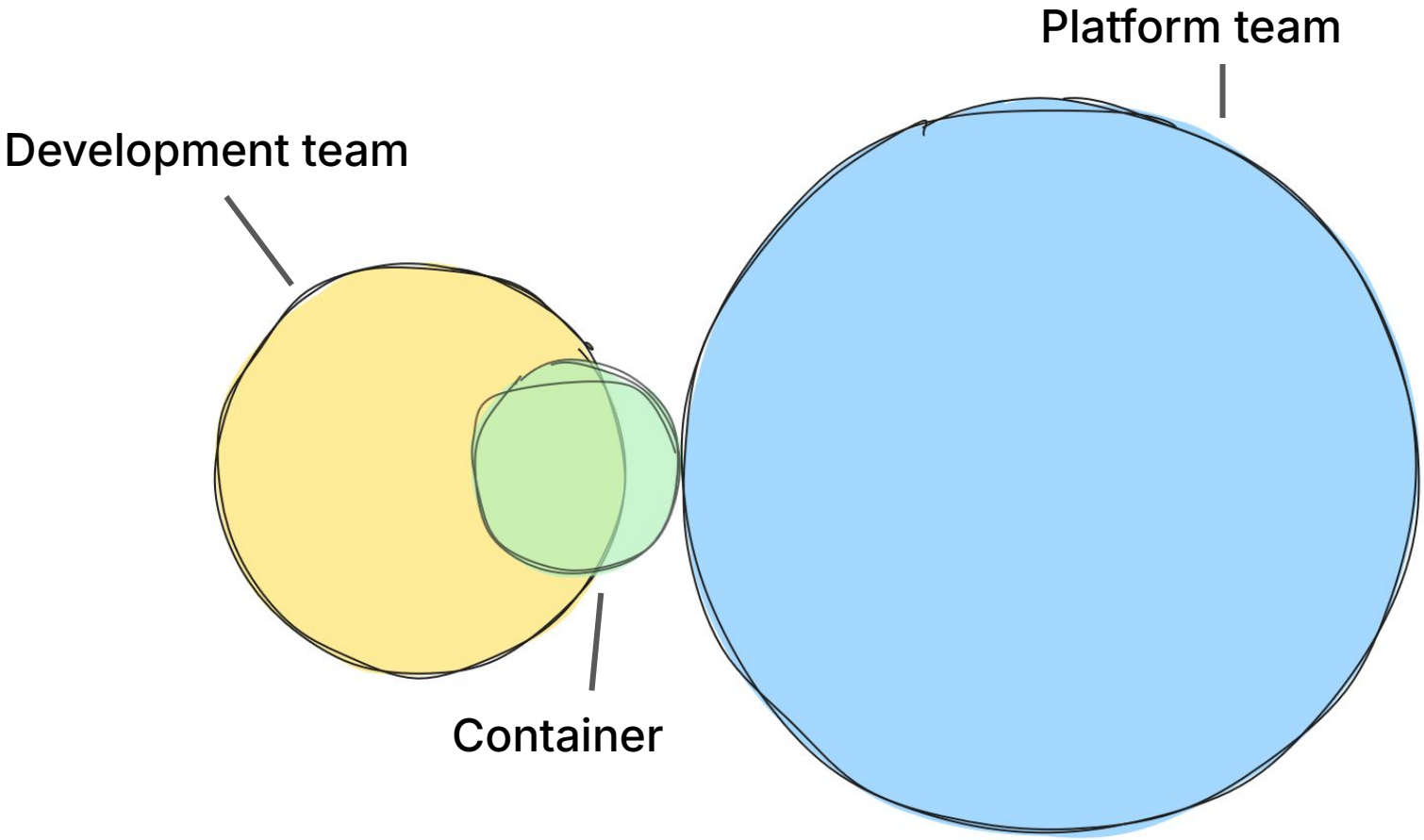
@erlendekern

Capra



@erlendekern

Capra



Inspired by devopstopologies.com

@erlendekern

Capra

@erlendekern

Capra

Dev 💥 Ops

@erlendekern

Capra

Serverless is a State of Mind

The point is focus — that is the why of serverless

Ben Kehoe Following 12 min read · Mar 17, 2019

@erlendekern

Capra

"Thinnest Viable Platform"

Real-world examples

The blueprint

Concluding thoughts

Erlend Ekern
AWS Community Builder
Cloud Architect

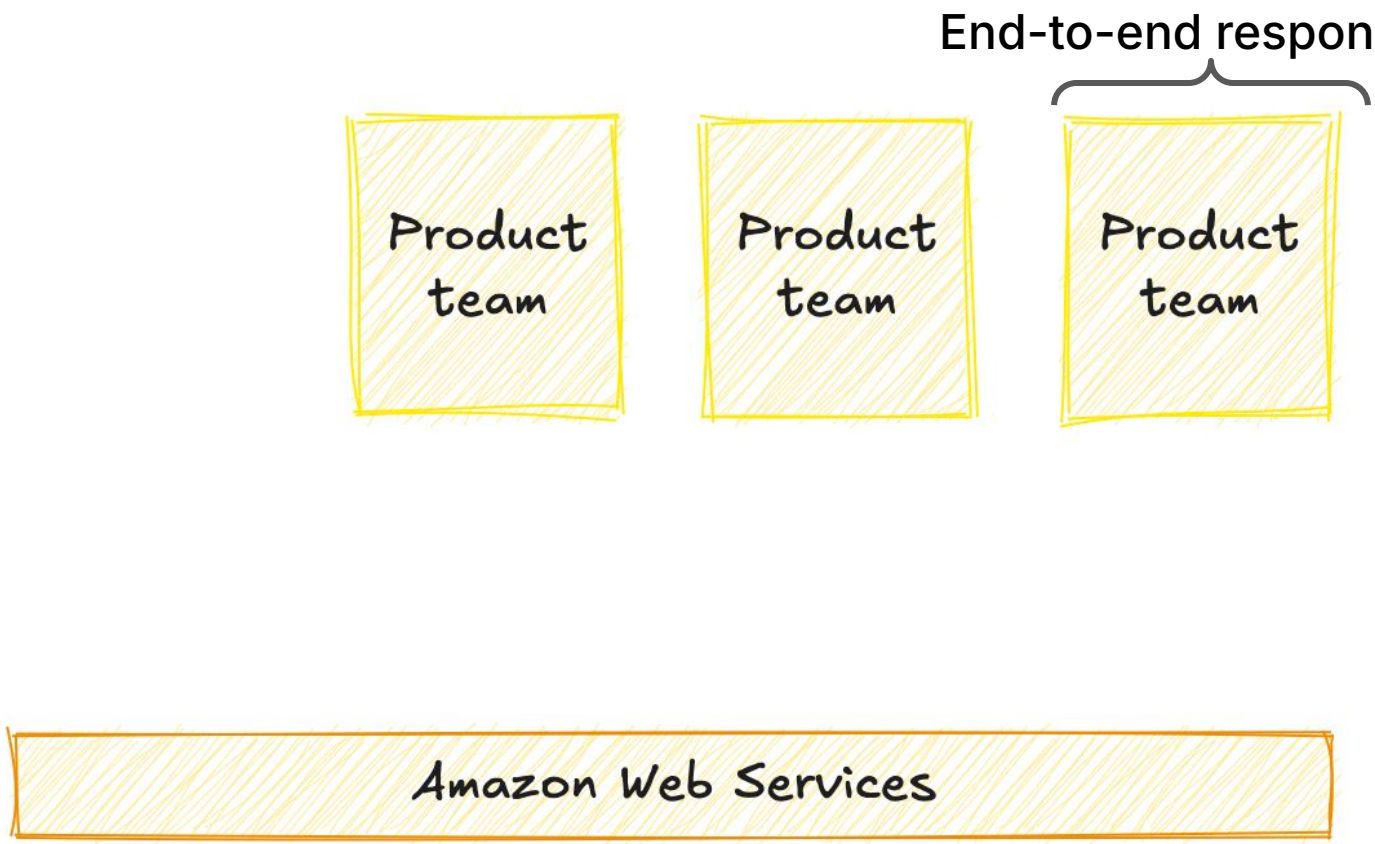
@erlendekern

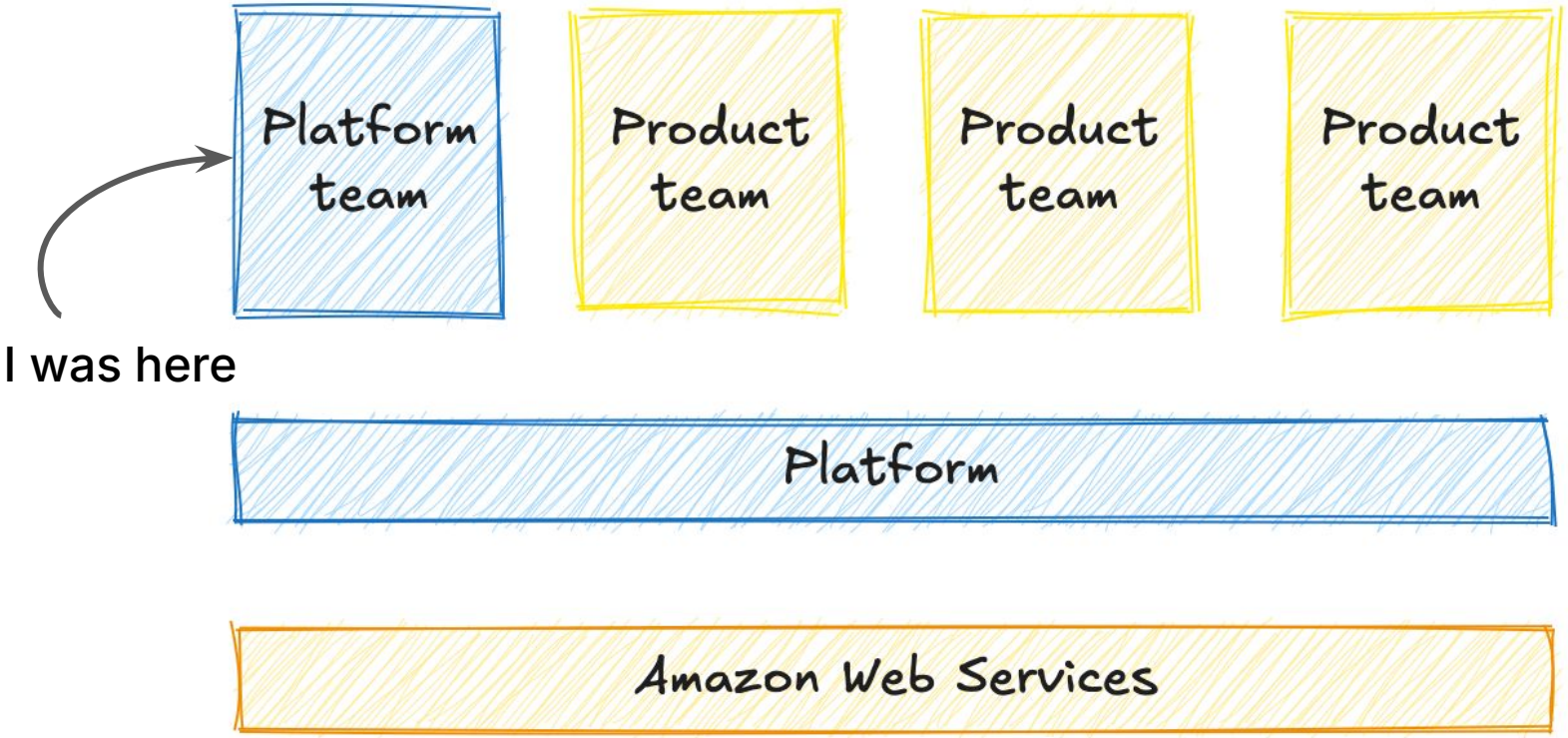
Capra

Liflig

@erlendekern

Capra


End-to-end responsible
Product team
Product team
Product team
Amazon Web Services

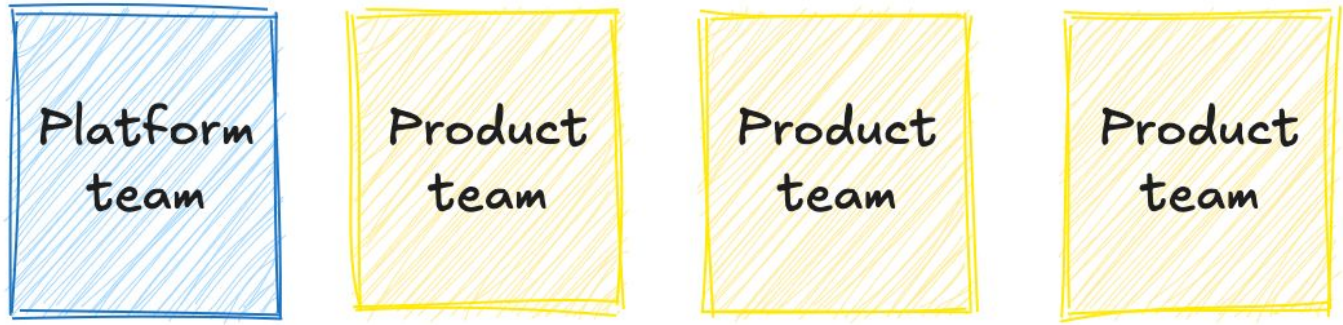


@erlendekern

Capra

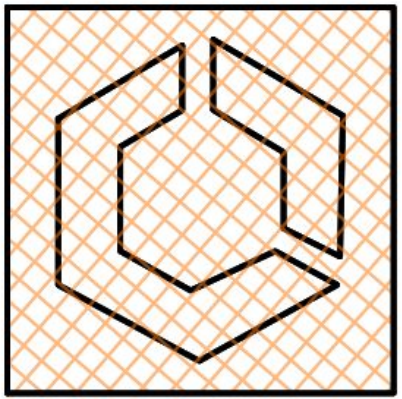
Hmm.. There's probably a managed service for this!

Applying a serverless mindset

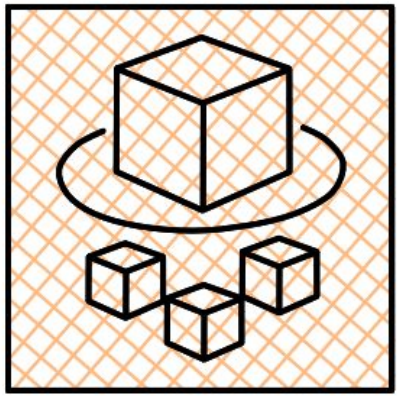


@erlendekern

Capra



ECS



Fargate

ECS

CloudFront

SNS

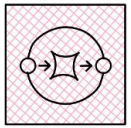
Aurora

Cognito

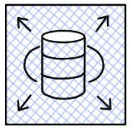
CloudWatch

Fargate

ALB



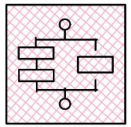
SQS



RDS

Lambda

API Gateway

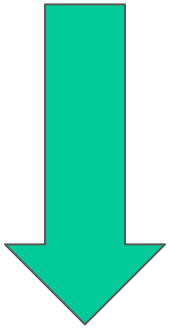


Step Functions

S3

@erlendekern

Capra



⚡ Onboarding to a production-ready foundation in **1 hour**

💪 Empowered teams that deployed **daily**

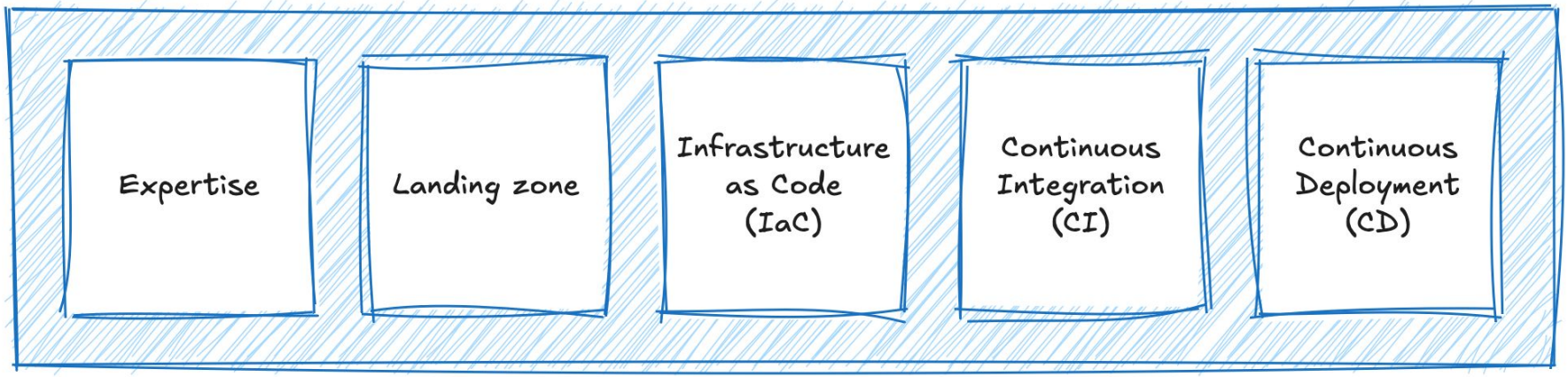
🧩 Lean platforms maintained by **2-3 people**

The blueprint

You build it, you run it!

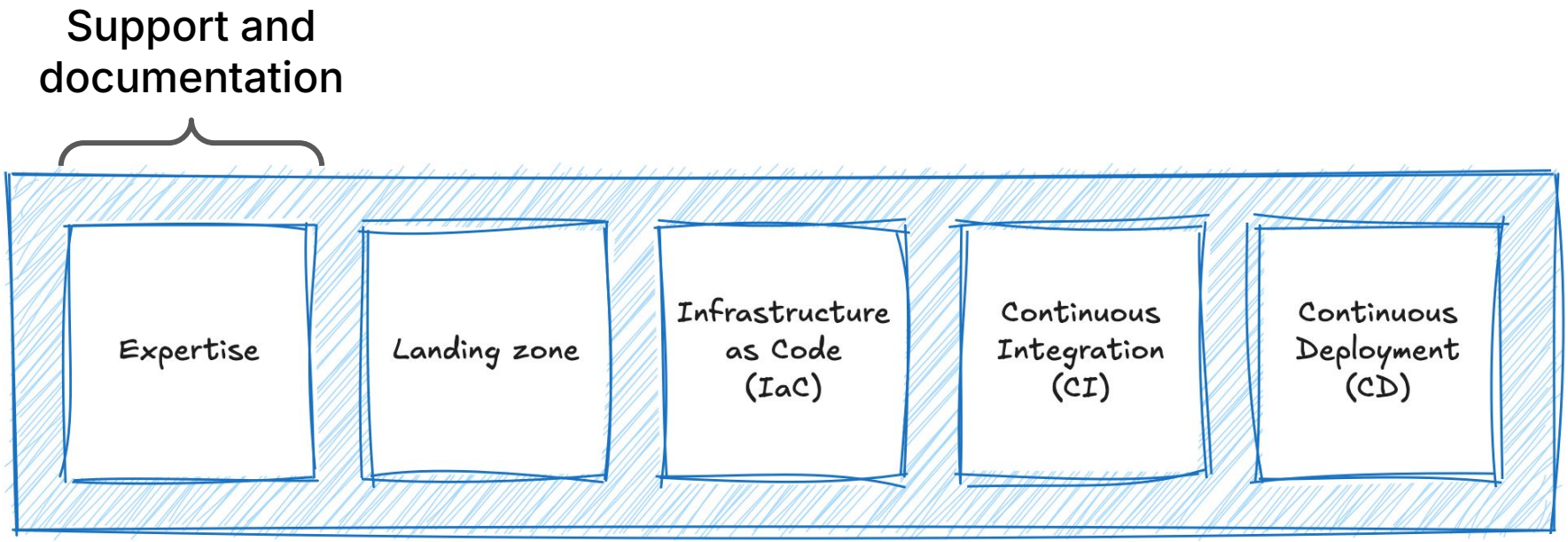
@erlendekern

Capra

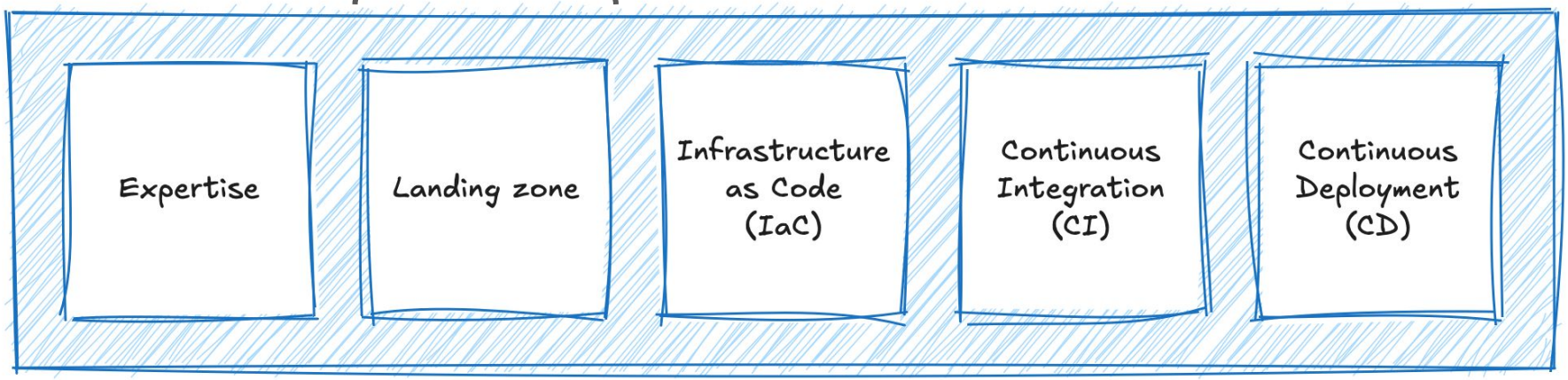


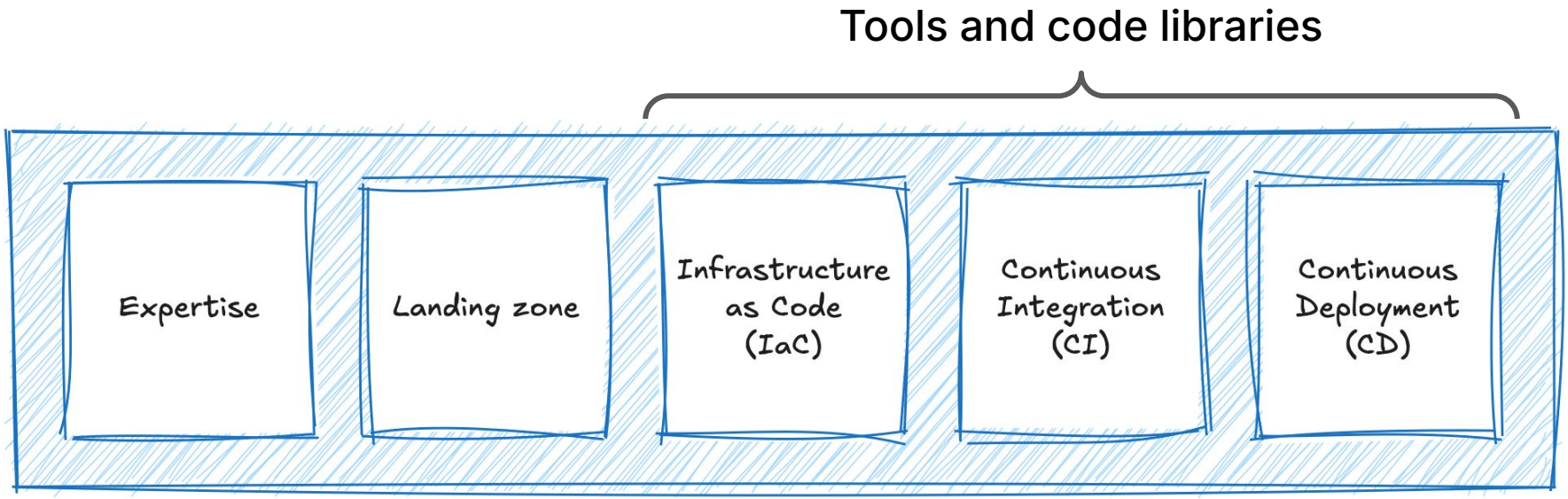
@erlendekern

Capra


Support and documentation
Expertise
Landing zone
Infrastructure as Code (IaC)
Continuous Integration (CI)
Continuous Deployment (CD)

Production-ready
cloud environments




Tools and code libraries
Expertise
Landing zone
Infrastructure as Code (IaC)
Continuous Integration (CI)
Continuous Deployment (CD)


Expertise

Expertise

@erlendekern

Capra

Expertise

https://www.lastweekinaws.com/blog/the-17-ways-to-run-containers-on-aws

@erlendekern

Capra


Landing zone

management

governance

Capra

Landing zone

"We might've overdone it with the negations..."

management

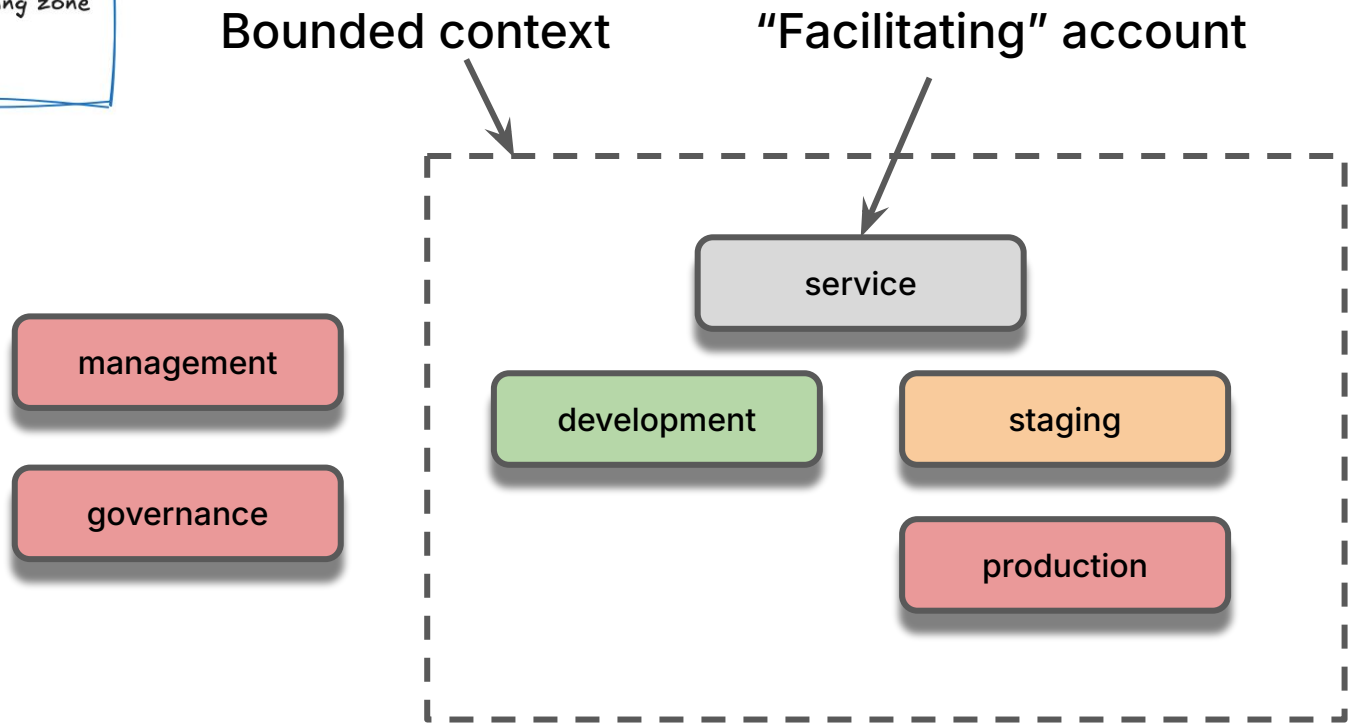
governance

```
{
  "Effect": "Deny",
  "NotAction": [ /* ... */],
  "Condition": {
    "StringNotEquals": { /* ... */  },
    "ArnNotLike": { /* ... */ }
  },
  "Resource": "*"
}
```

@erlendekern

Capra

Landing zone



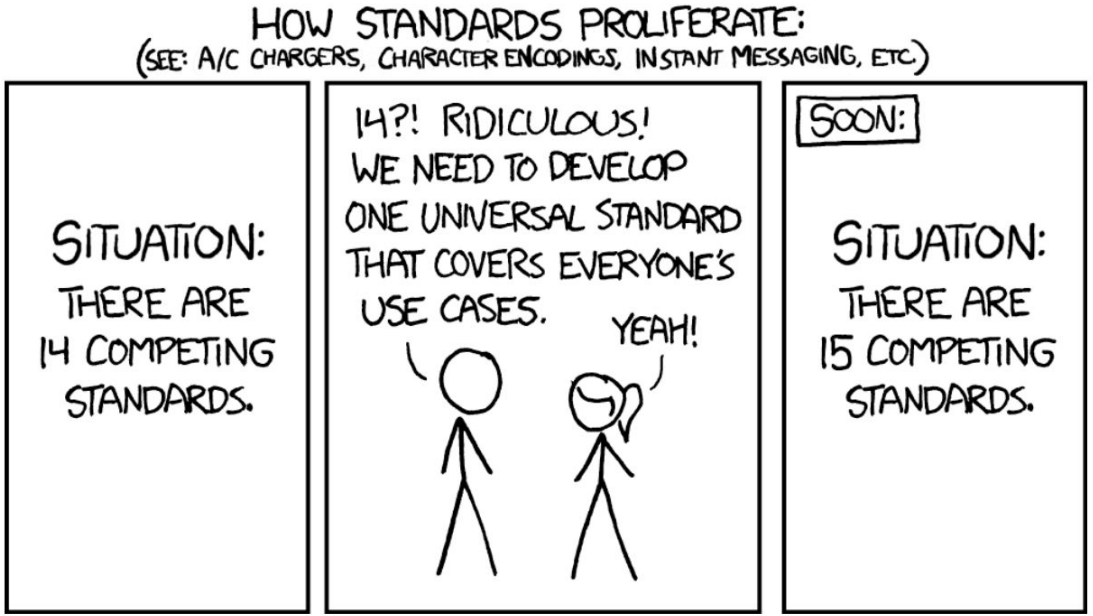
@erlendekern

Capra

“AWS Control Tower offers a straightforward way to set up and govern an AWS multi-account environment”

“AWS Control Tower offers a ~~straightforward~~ way to set up and govern an AWS multi-account environment”

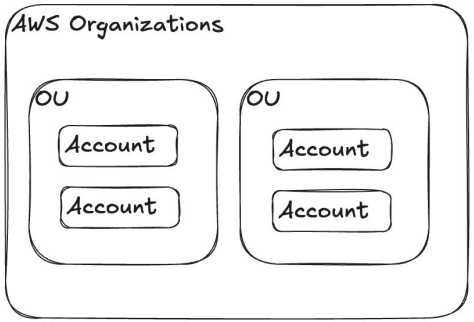
AWS Landing Zone
AWS Control Tower
AWS Account Factory for Terraform
AWS Landing Zone Accelerator
org-formation
superwerker
++

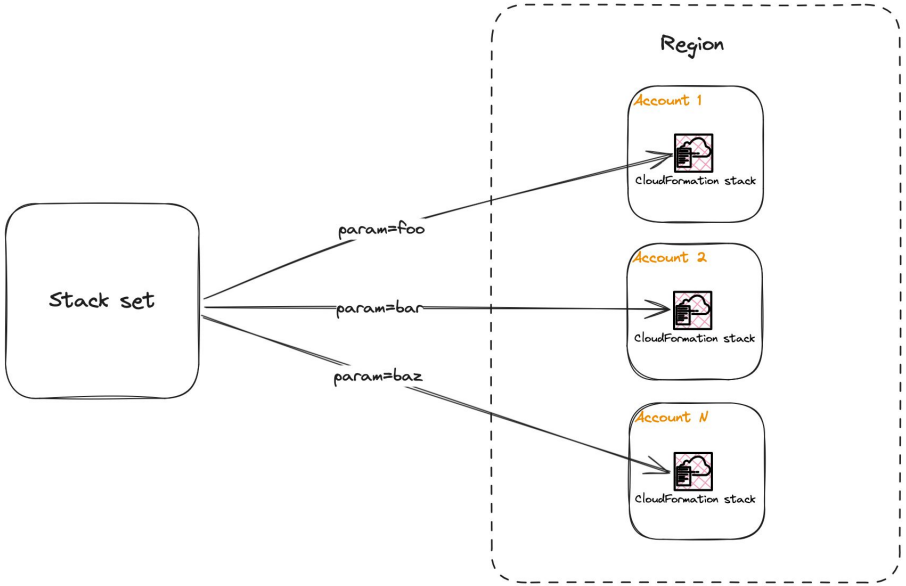


https://xkcd.com/927

@erlendekern

Capra





org management

account baselines

Terraform + CloudFormation StackSets?

@erlendekern

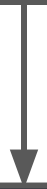
Capra


```
  # module.account["aws+developer+sandbox@example.com"].aws_organizations_account.this will be created
  + resource "aws_organizations_account" "this" {
      + arn                       = (known after apply)
      + close_on_deletion         = true
      + create_govcloud           = false
      + email                     = "aws+developer+sandbox@example.com"
      + govcloud_id               = (known after apply)
      + iam_user_access_to_billing = "ALLOW"
      + id                        = (known after apply)
      + joined_method             = (known after apply)
      + joined_timestamp          = (known after apply)
      + name                      = "developer-sandbox"
      + parent_id                 = "ou-o8e9-ldpgc8p5"
      + role_name                 = "OrganizationAccountAccessRole"
      + status                    = (known after apply)
      + tags_all                  = {
          + "managed" = "true"
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

@erlendekern

Capra


```
import { /* ... */ }

moved  { /* ... */ }
```



```
No changes. Your infrastructure matches the configuration.
```

@erlendekern

Capra


```yaml
# yaml-language-server: $schema=./organization.schema.json
organization:
  id: "o-laso1x2apk"
  organizational-units:
    security:
      accounts:
        - name: "security"
          environment: "prod"
          email: "aws+security+prod@example.com"
          owner: "team-developer-platform"
    workloads:
      accounts:
        - name: "foobar-dev"
          environment: "dev"
          email: "aws+foobar+dev@example.com"
          owner: "team-foxtrot"
        - name: "foobar-staging"
          environment: "dev"
          email: "aws+foobar+dev@example.com"
          owner: "team-foxtrot"
    sandbox:
      accounts:
        - name: "developer-sandbox"
          environment: "sandbox"
          email: "aws+developer+sandbox@example.com"
          owner: "team-echo"
```

@erlendekern

Capra


```hcl
data "aws_organizations_organization" "this" {}

locals {
  org_config        = yamldecode(file("../organization.yml")).organization
  accounts_by_email = {} # Omitted HCL monstrosity ...
}

resource "aws_organizations_organization" "this" {
  # Configure enabled organizational services, etc.
}

resource "aws_iam_organizations_features" "this" {
  # Enable centralized root user, etc.
}

resource "aws_organizations_organizational_unit" "ou" {
  for_each  = local.org_config.organizational-units
  name      = each.key
  parent_id = data.aws_organizations_organization.this.roots[0].id
}

resource "aws_organizations_account" "this" {
  for_each                  = local.accounts_by_email
  email                     = each.key
  name                      = each.value.name
  organizational_unit_id    = aws_organizations_organizational_unit.ou[each.value.ou].id
  role_name                 = "OrganizationAccountAccessRole"
  close_on_deletion         = true
  iam_user_access_to_billing = "ALLOW"
  lifecycle {
    ignore_changes = [name, email, role_name, iam_user_access_to_billing]
  }
}
```

Parse file

Create resources

@erlendekern

Capra

blog.ekern.me

Feb 27, 2025

What's the deal with AWS CloudFormation StackSets?

Account baselines

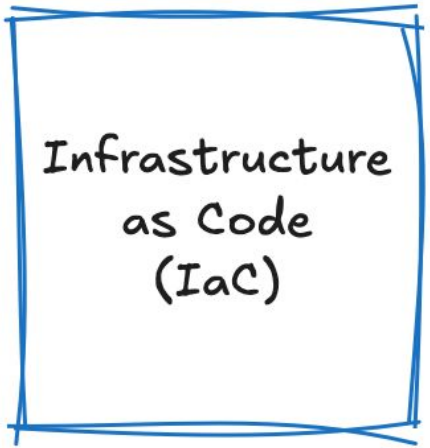
```hcl
resource "aws_cloudformation_stack_set" "this" {
  for_each = {
    "critical-activity-monitoring" = "../assets/templates/cfn-critical-activity-monitoring.yml"
    "cdk-bootstrap"                = "../assets/templates/cfn-cdk-bootstrap.yml"
    "dns-zone"                     = "../assets/templates/cfn-dns-zone.yml"
  }
  name             = each.key
  permission_model = "SERVICE_MANAGED"
  call_as          = "DELEGATED_ADMIN"
  auto_deployment {
    enabled                          = true
    retain_stacks_on_account_removal = true
  }
  template_body = file(each.value)
}

resource "aws_cloudformation_stack_set_instance" "cdk_bootstrap" {
  for_each       = aws_cloudformation_stack_set.this
  stack_set_name = each.value.name
  region         = "eu-west-1"
  call_as        = "DELEGATED_ADMIN"
  deployment_targets {
    organizational_unit_ids = local.ous_by_name["workloads"].id
  }
}
```

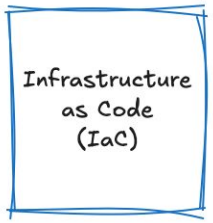
@erlendekern

Capra

primeharbor / **org-kickstart**

cdklabs / **cdk-stacksets**


Infrastructure
as Code
(IaC)



AWS CloudFormation

Terraform

Serverless Framework

AWS SAM

Pulumi

AWS CDK

cdktf

SST

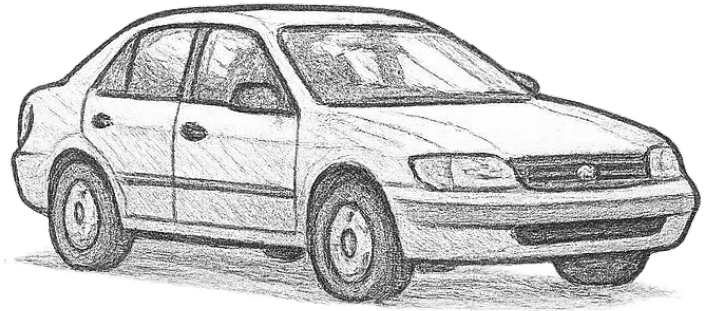
Alchemy

???

@erlendekern

Capra

Infrastructure as Code (IaC)

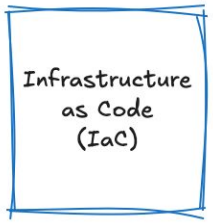


Terraform

AWS CDK

@erlendekern

Capra



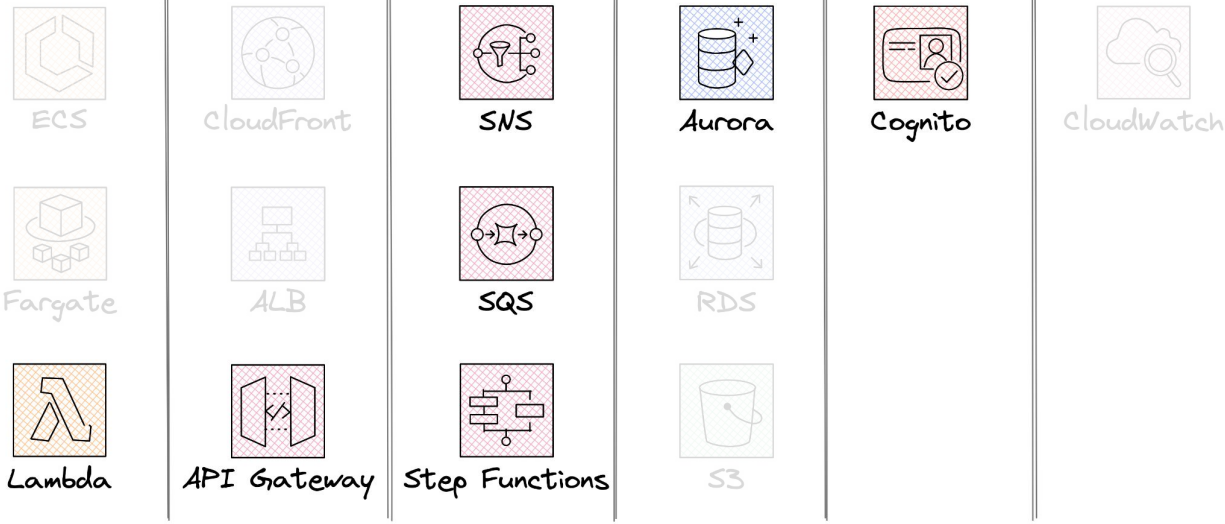
Thin **library** on top of AWS primitives

Infrastructure as Code (IaC)

@erlendekern

Capra

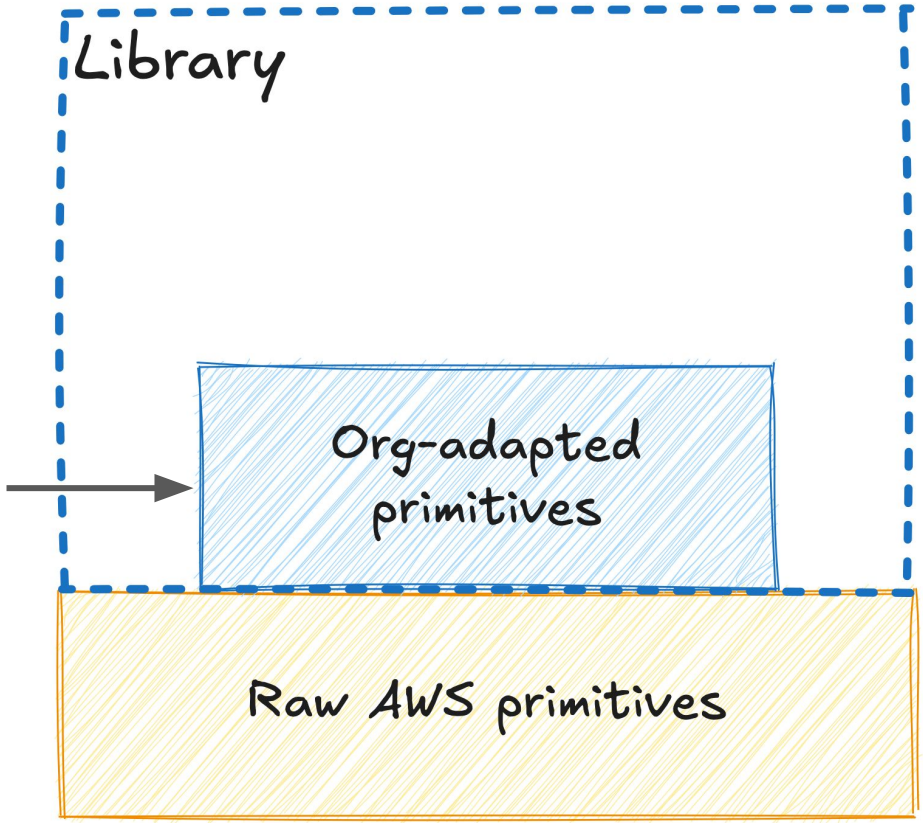
Infrastructure as Code (IaC)



@erlendekern

Capra

Infrastructure as Code (IaC)

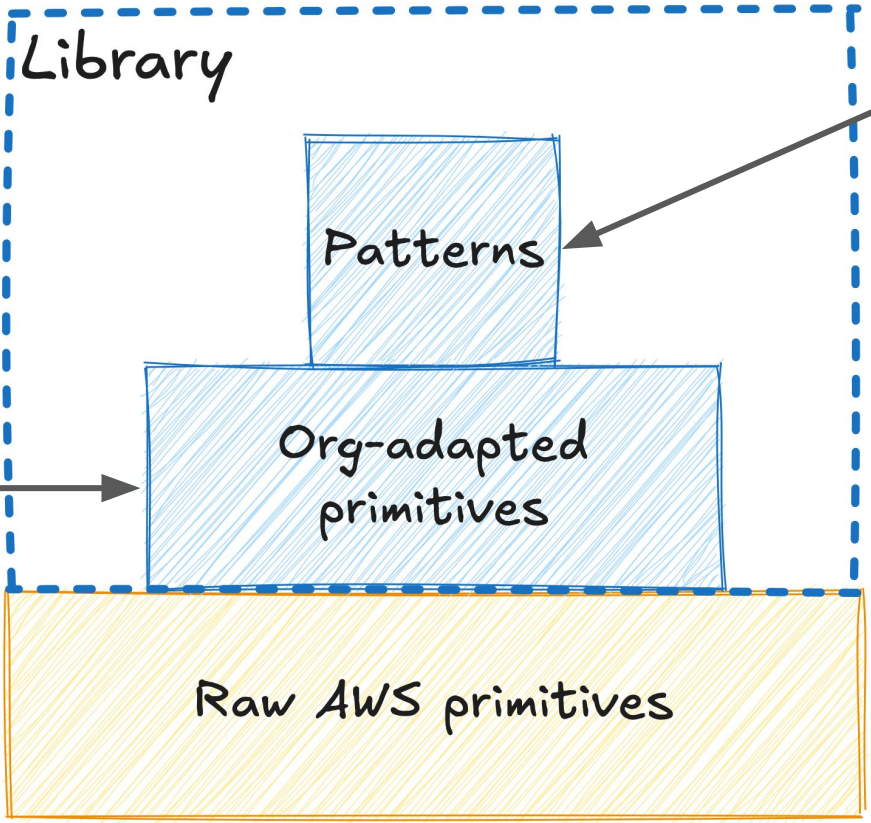


Focus on these

@erlendekern

Capra

Infrastructure as Code (IaC)



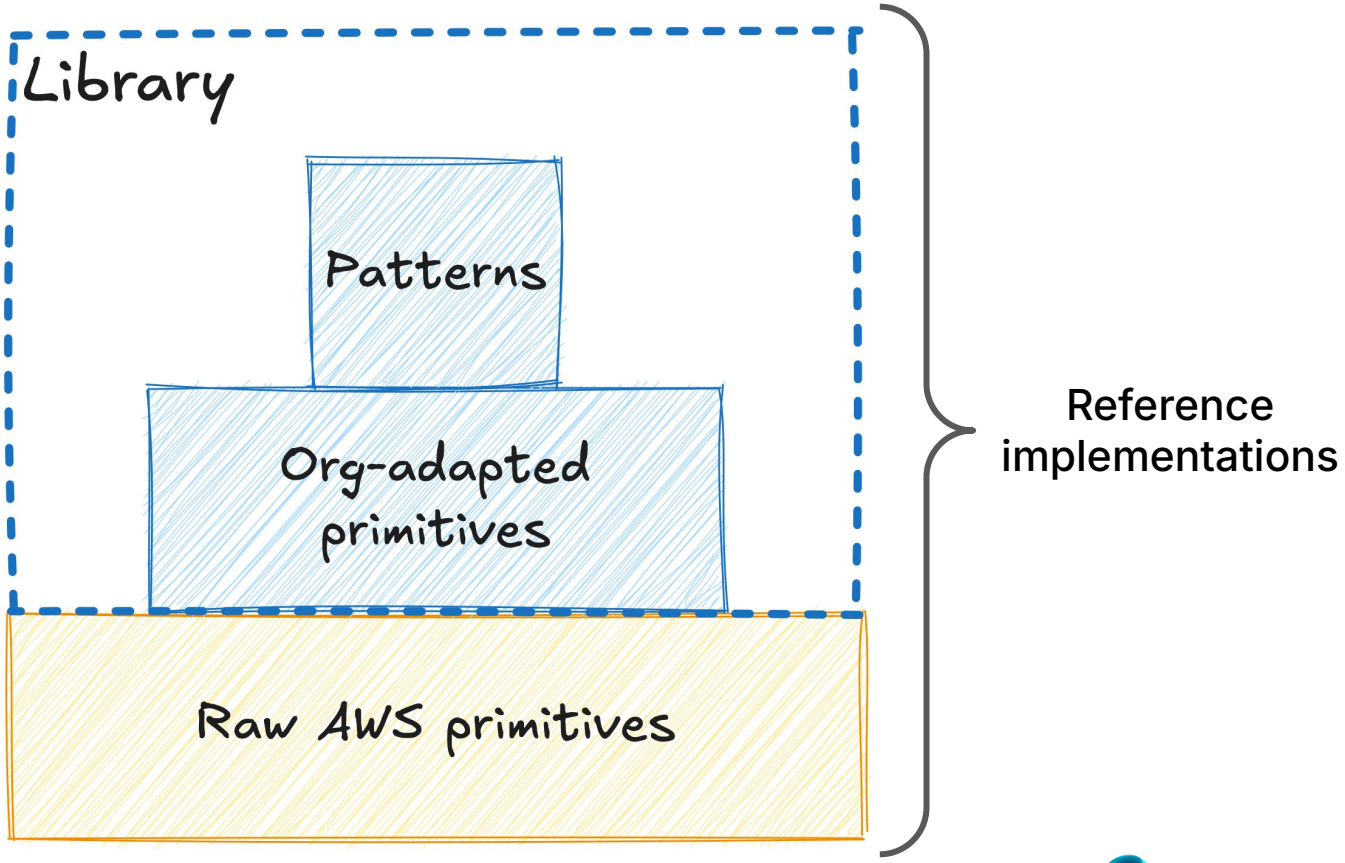
A few of these

Focus on these

@erlendekern

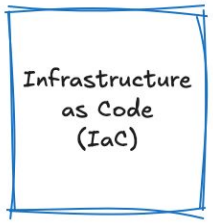
Capra

Infrastructure as Code (IaC)

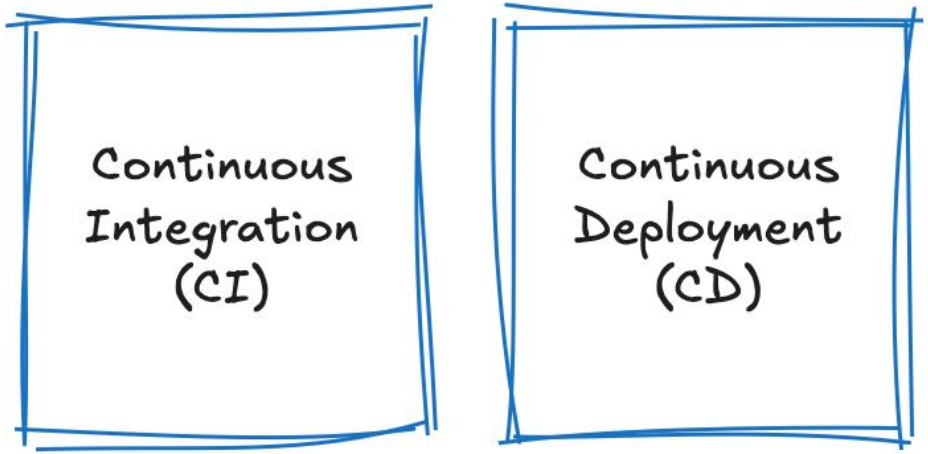


@erlendekern

Capra

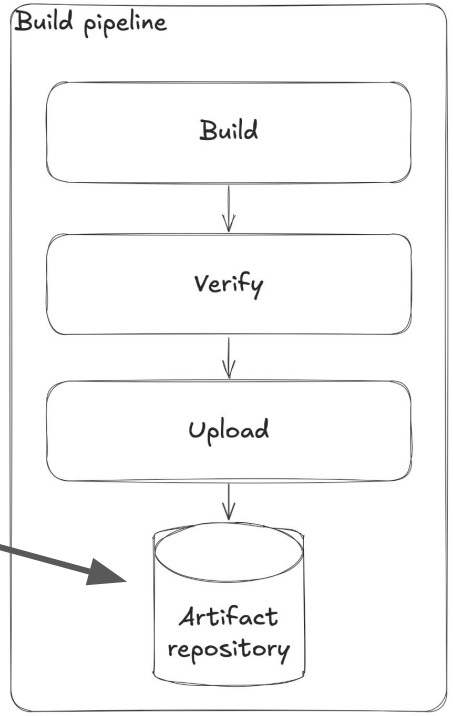


A forcing function for expertise?


Continuous
Integration
(CI)
Continuous
Deployment
(CD)

Continuous Integration (CI)

Continuous Deployment (CD)



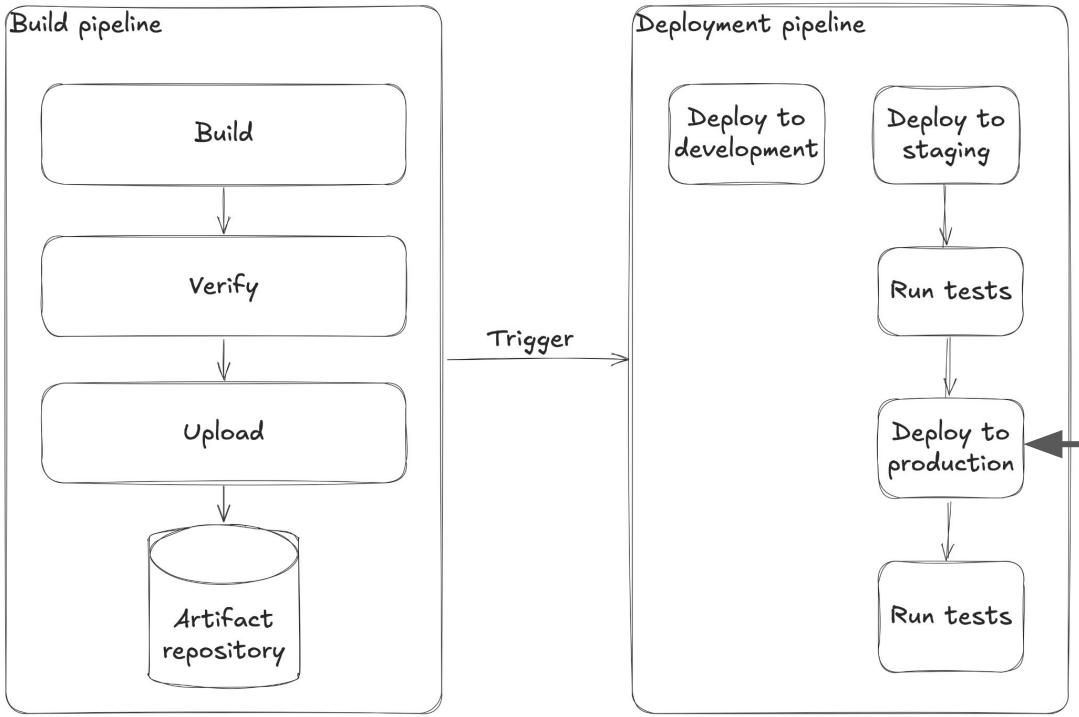
Build-once-deploy-many

@erlendekern

Capra

Continuous Integration (CI)

Continuous Deployment (CD)




```
$ terraform apply
$ pulumi up
$ cdk deploy
```

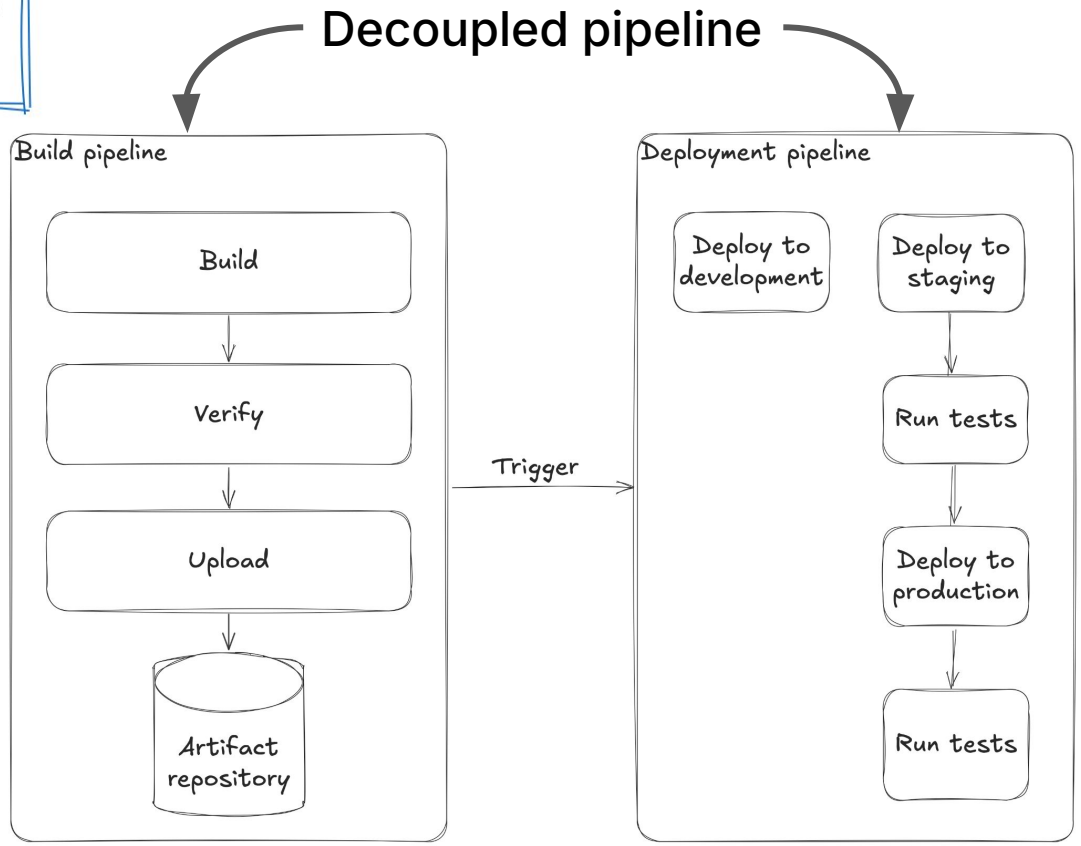
App deploy through infra

@erlendekern

Capra

Continuous Integration (CI)

Continuous Deployment (CD)



@erlendekern

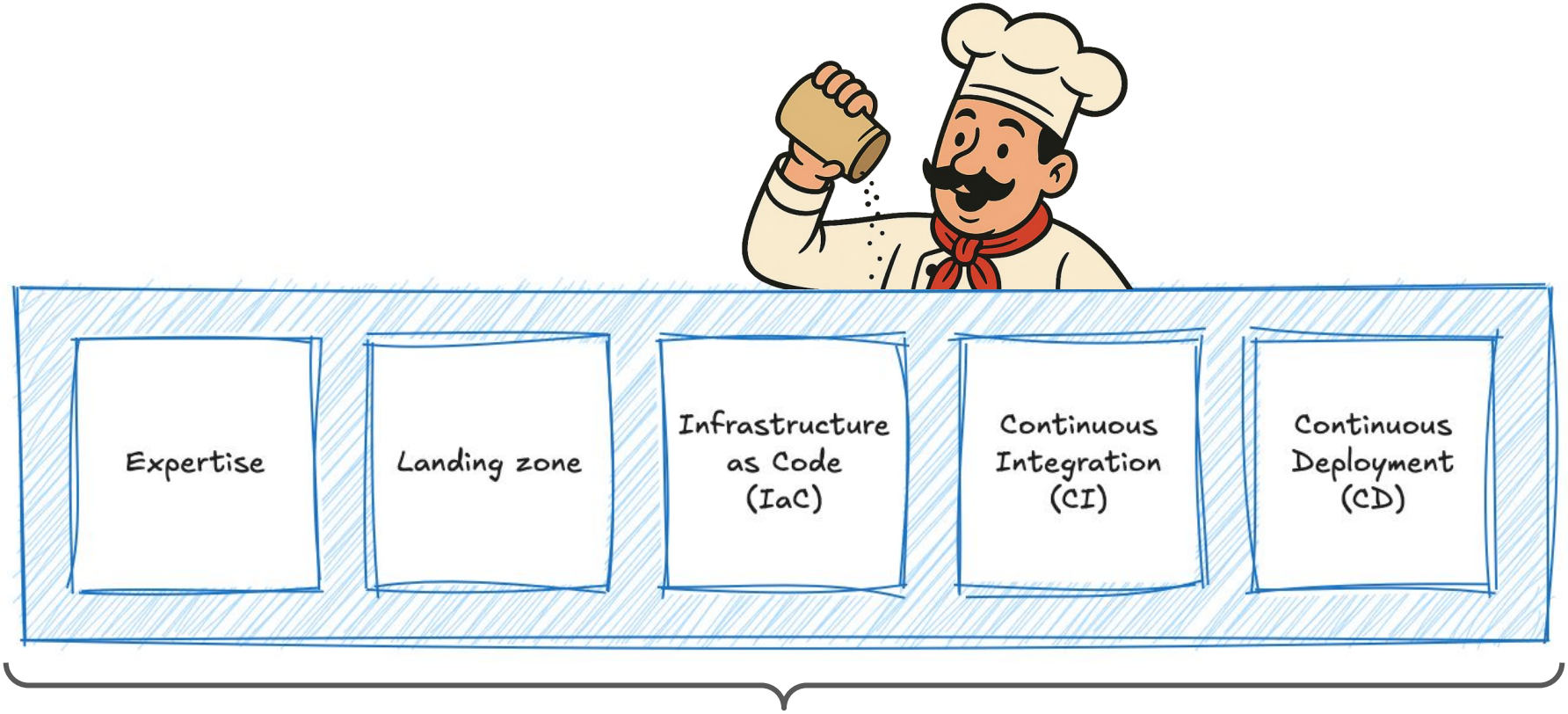
Capra

Continuous Integration (CI)

Continuous Deployment (CD)

@erlendekern

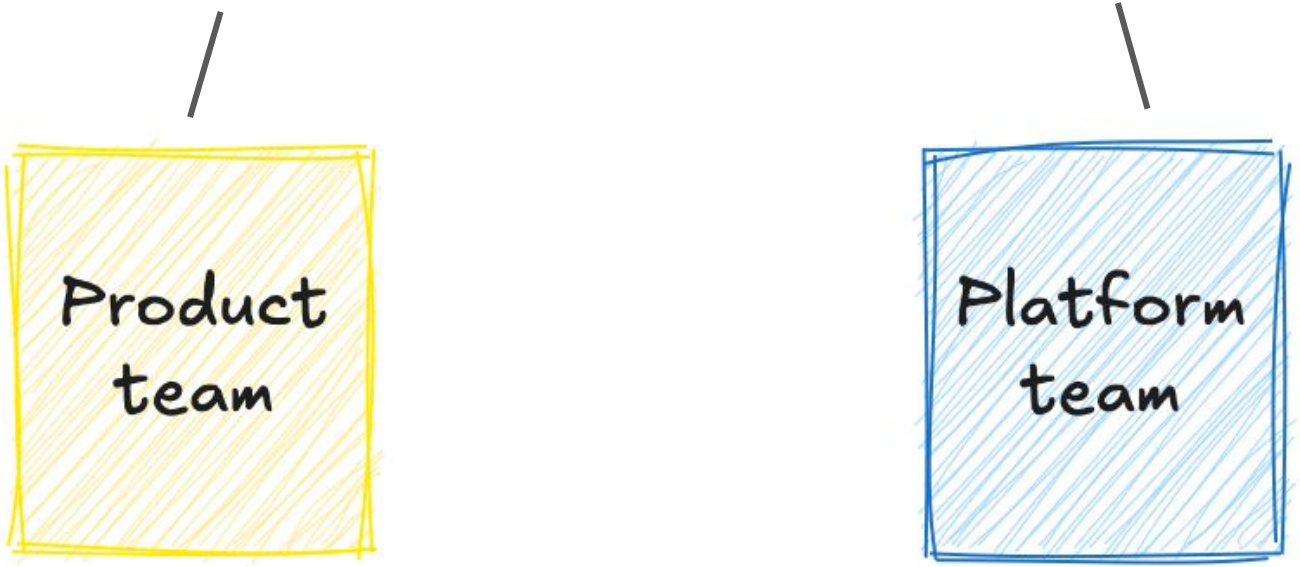
Capra


Expertise
Landing zone
Infrastructure as Code (IaC)
Continuous Integration (CI)
Continuous Deployment (CD)

Starter kit

Hey, could you help us out with cloud environments for *foobar*?

Sure can do!



Hi 👋

The AWS account set for bounded context **foobar** has been created, and the accounts are ready to use:

- *foobar-service* `123456789012`
- *foobar-dev* `234567890123`
- *foobar-staging* `345678901234`
- *foobar-prod* `456789012345`

Each account has its own DNS zone under `foobar.example.com` (e.g., `prod.foobar.example.com`).

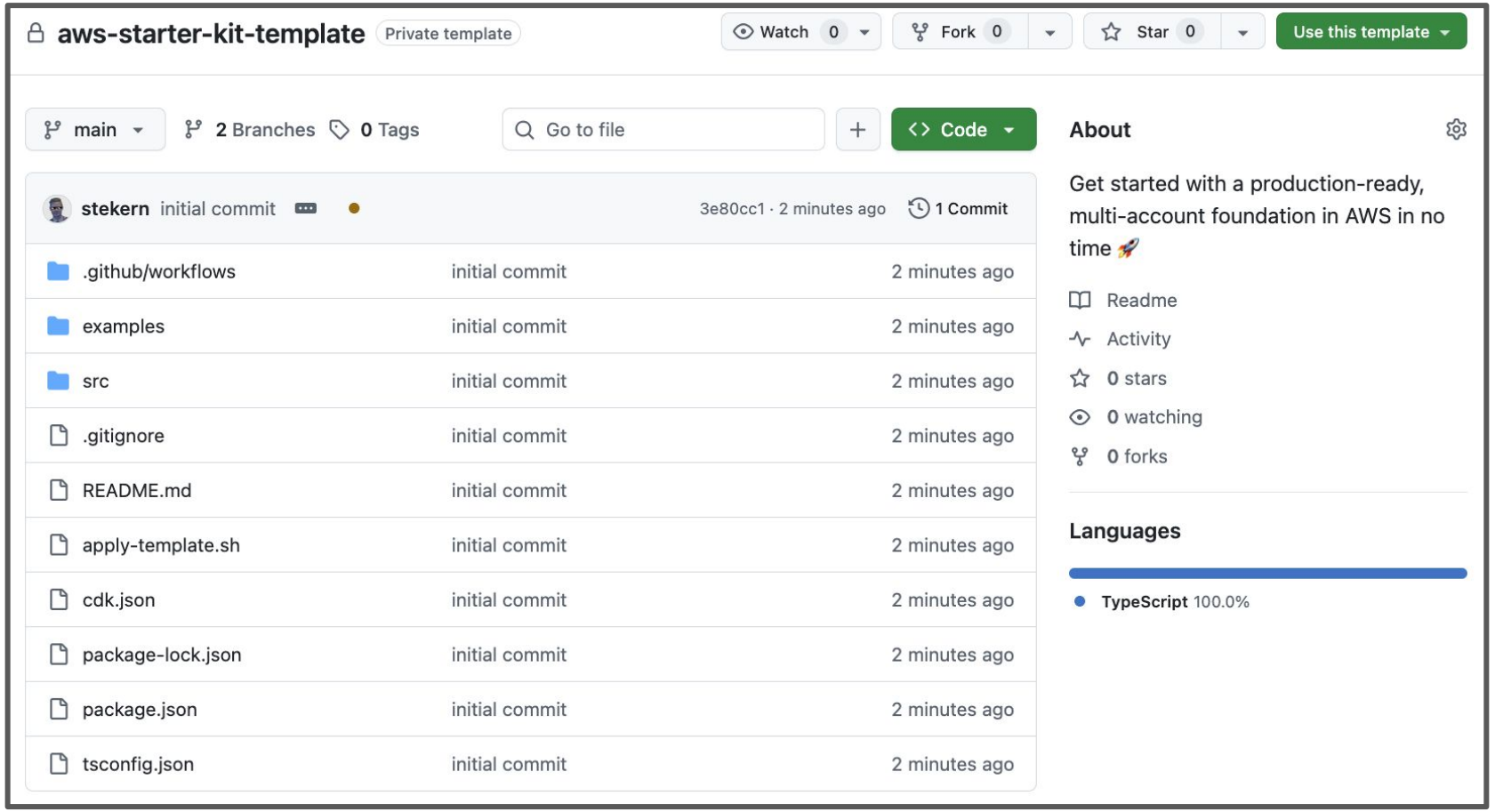
Check out our Getting started guide to configure access to AWS and get started with your very own Infrastructure as Code (IaC) repository.

You'll be up and running on a production-ready, multi-account foundation in no time 🚀

If you have any questions at all, don't hesitate to reach out in #platform-support!

@erlendekern

Capra



@erlendekern

Capra

```ts
// src/config.ts

export const accounts = {
  service: "<service-account-id>",
  dev: "<dev-account-id>",
  staging: "<staging-account-id>",
  prod: "<prod-account-id>",
}

export const defaultRegion = "<default-aws-region>"
export const boundedContext = "<bounded-context>"
export const dnsZone = "<dns-zone>"

export const artifactBucketName = `${boundedContext}-starter-kit-artifact-${accounts.service}-${defaultRegion}`
export const ecrRepositoryName = `${boundedContext}-starter-kit-artifact`

export const trustedRepositories = [
  {
    name: "<repository-name>",
    owner: "<repository-owner>",
    branches: ["<repository-default-branch>"],
  },
]
```

Usage

1. Create a new repository using this template by clicking here: ✨ Use template ✨
2. Name the repository using the following convention `<bounded-context>-infra` (e.g., `example-infra`).
3. Click **Create repository**, clone the repository to your local machine, and follow the steps below 🔄
4. Install the npm packages: `$ npm ci`
5. Log in to the `service` account of the AWS account set: `$ assume <bounded-context>-service-admin`
6. Replace placeholders and create the deployment pipeline by running the `apply-template.sh` script:

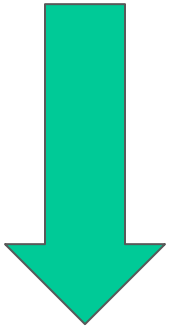
   ▸ **Show option details and example values**

```
$ ./apply-template.sh                          \
  --bounded-context     "<bounded-context>"    \
  --dns-zone            "<dns-zone>"           \
  --default-aws-region  "<default-aws-region>" \
  --service-account-id  "<service-account-id>" \
  --dev-account-id      "<dev-account-id>"     \
  --staging-account-id  "<staging-account-id>" \
  --prod-account-id     "<prod-account-id>"
```

7. Commit your changes and push them to trunk
8. Let the CI/CD pipeline take care of the rest 😎

@erlendekern

Capra



- ★ Their own IaC codebase covering all environments
- ★ Their own deployment pipelines, ready-to-go
- ★ Full access to their AWS accounts
- ★ DNS zones and TLS certificates
- ★ A demo application in each environment
- ★ Documentation on where to go next

The perfect platform?

No, but an **effective** one

@erlendekern

Capra

Challenges

- Skills and maturity
- Ownership boundaries
- Variation and consistency
- Shadow platforming

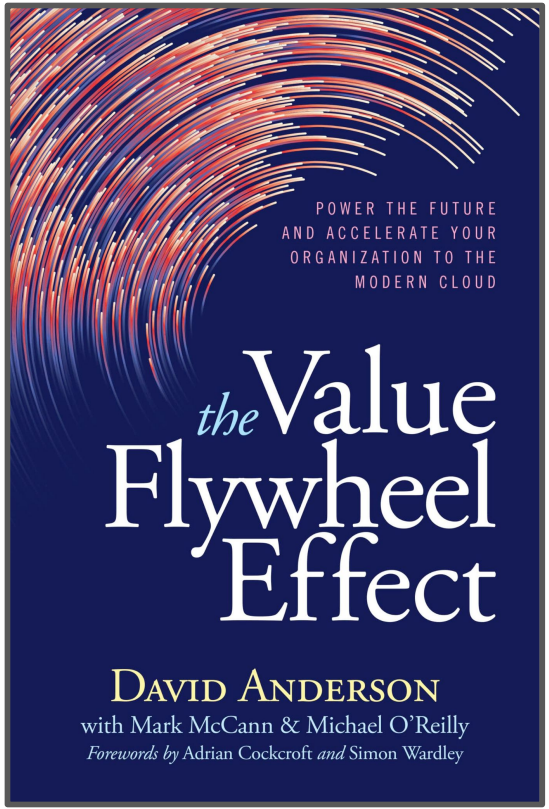
tl;dr

Enablement > restriction

Outcomes > technologies

Constraints > flexibility

Primitives > large abstractions



@erlendekern

Capra

Thank you!

🔗 ekern.me
🔗 linkedin.com/in/erlendekern
🔗 twitter.com/erlendekern

@erlendekern

Capra