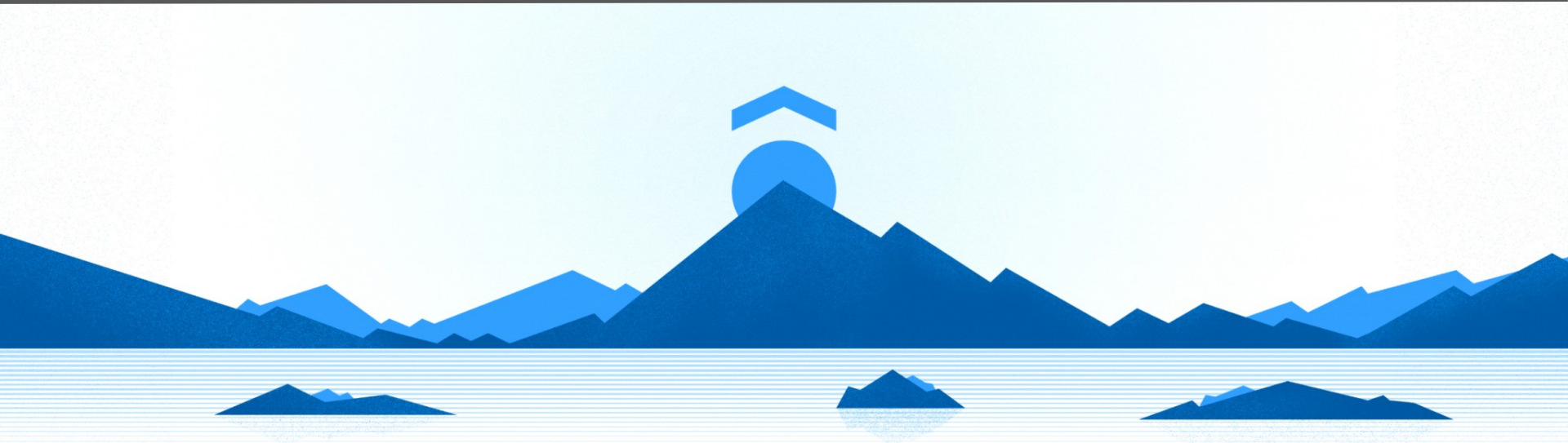
ZERO

Opinionated infrastructure to take you from idea to production on day one



Commit

The remote-first developer community where startup engineers get paid to find their next career opportunity

Bill Monkman

Chief Architect

Specializes in Backend, DevOps, Distributed Systems, Scaling SaaS startups

Why did we build Zero?

Startups…

- Are constantly reinventing the wheel
- Have a DevOps knowledge gap
- Are often focused on customer value but not foundational software
- Often have to re-architect a couple years in

How does it help?

Set up all the necessary components

Built-in best practices
- Security, Scalability, Availability, Visibility, 12 Factor, etc.

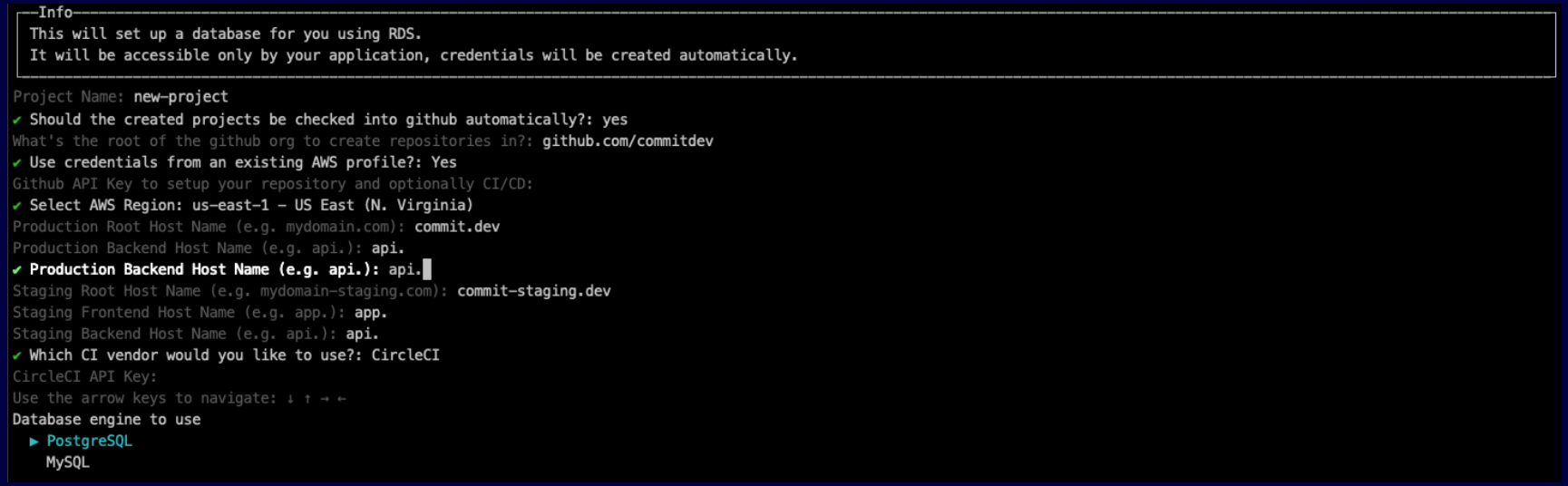
Learning

Ownership

How does it work?

`zero init`

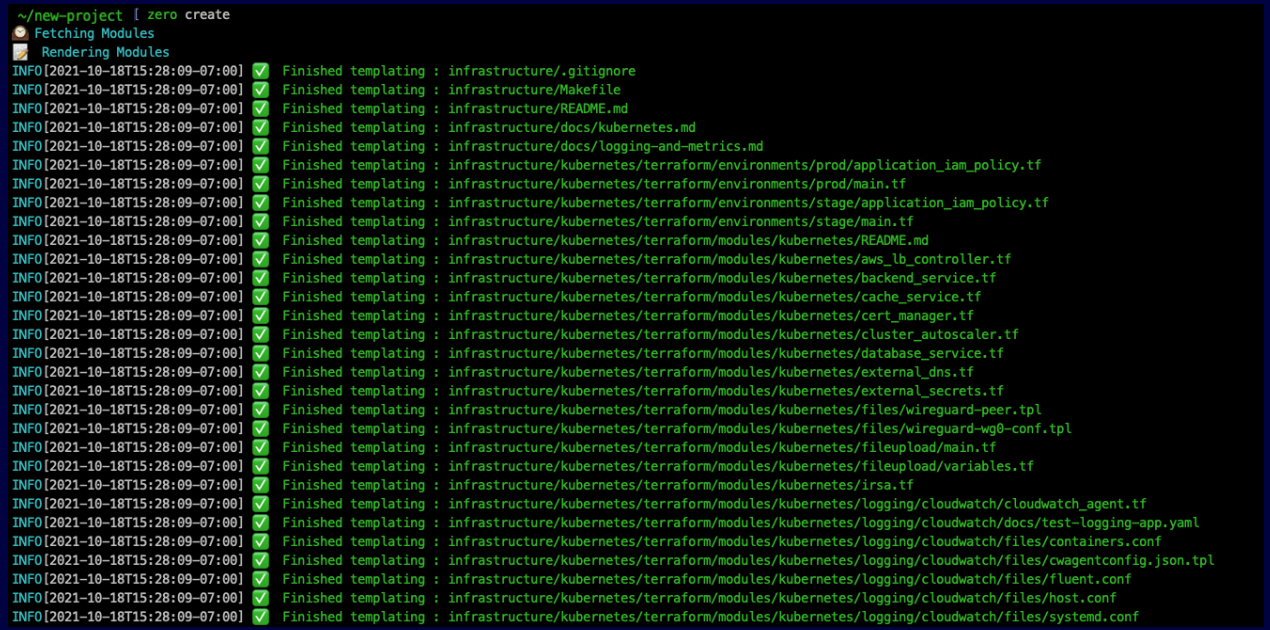
Interview phase



How does it work?

`zero create`

Code creation phase



How does it work?

`zero apply`

Infrastructure creation phase

```
 ~/new-project/new-project [ zero apply
Choose the environments to apply. This will create infrastructure, CI pipelines, etc.
At this point, real things will be generated that may cost money!
Only a single environment may be suitable for an initial test, but for a real system we suggest setting up both staging and production environments.
✔ Staging
🔍  checking project new-project's module requirements.
Running command check
Successfully found binary(s): gh
true
Running command check
Successfully found binary(s): gh
true
🎉  Bootstrapping project new-project. Please use the zero-project.yml file to modify the project as needed.
Cloud provider: AWS
Runtime platform: Kubernetes
Infrastructure executor: Terraform
Executing apply command for zero-aws-eks-stack...
cd /Users/bill/new-project/new-project/infrastructure && AUTO_APPROVE="-auto-approve" make
aws s3 ls new-project-stage-terraform-state > /dev/null 2>&1 || ( \
        cd terraform/bootstrap/remote-state && \
        terraform init && \
        terraform apply -var "environment=stage" -auto-approve && \
        rm ./terraform.tfstate )

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v3.50.0

Terraform has been successfully initialized!
```

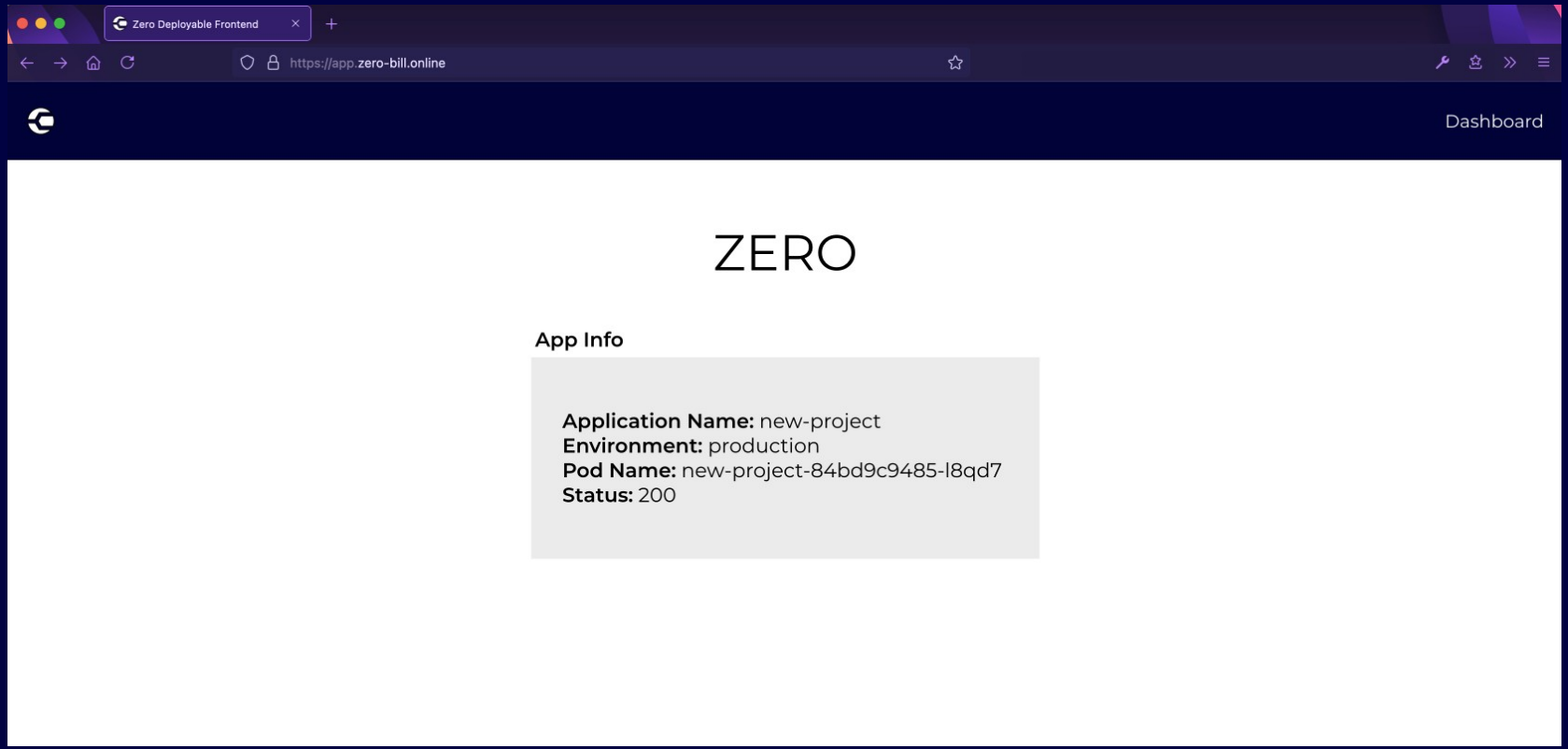
How does it work?

`zero init`
Interview phase

`zero create`
Code creation phase

`zero apply`
Infrastructure creation phase

How does it work?



What are the components?

Infrastructure

Backend

Frontend

Marketing Site

CI/CD to tie everything together

What are the components?

Infrastructure

AWS
EKS *-or-* Serverless using SAM (upcoming)
- Nginx Ingress Controller
- External DNS
- Cert Manager
- Cluster Autoscaler
- Ory Kratos + Oathkeeper
- External Secrets

RDS
Logging via Cloudwatch *-or-* Elasticsearch + Kibana
Metrics via Cloudwatch *-or-* Prometheus + Grafana
VPN w/ Wireguard

What are the components?

Backend

Golang *-or-* Node.js

Deployed into EKS

Application features

- User management / login / authentication
- Billing w/ Stripe
- File upload / download using Signed CloudFront URLs
- Notifications via Email / SMS / Slack

What are the components?

Frontend

React

Deployed into S3, served from CloudFront

Application features

- User management / login / authentication
- Billing w/ Stripe
- File upload / download using Signed CloudFront URLs
- Notifications via Email / SMS / Slack

What are the components?

Marketing Site / Landing Page

Static site generation w/ Gatsby

What are the components?

CI/CD

CircleCI *-or-* GitHub Actions

- Running tests
- Linting
- Deployment to staging and production environments

What’s it like to use it?

All the code and infrastructure is yours

Automated deployments

Focus on Developer Experience

Learning Resources

Support - Community

Ory Integration

User management / authentication goals

No more writing user management code

Unified authentication checking for backends

Represented as much as possible in Terraform

Ory Integration

Kratos and Oathkeeper

Kubernetes Operator (Maester)
Helm Terraform Provider
Config merging for flexibility
Application-controlled config at deploy time

Ory Integration

`user_auth` module (Terraform Registry) allows us to easily install and configure both Kratos and Oathkeeper

```
user_auth = [
  {
    name                          = local.project
    auth_namespace                = "user-auth"
    kratos_secret_name            = local.project
    frontend_service_domain       = "app.${local.domain_name}"
    backend_service_domain        = "api.${local.domain_name}"
    whitelisted_return_urls       = ["https://api.${local.domain_name}"]
    jwks_secret_name              = "${local.project}-${local.environment}-oathkeeper-jwks-${local.random_seed}"
    # This domain or address must be verified by the mail provider (Sendgrid, SES, etc.)
    user_auth_mail_from_address   = "noreply@${local.domain_name}"
    kratos_values_override        = {}
    oathkeeper_values_override    = {}
  }
```

Ory Integration

All config values can be overridden

```
module "kratos_config" {
  source  = "cloudposse/config/yaml"
  version = "0.7.0"

  map_config_local_base_path = "${path.module}/files"
  map_config_paths           = ["kratos-values.yml"]
  map_configs                = [local.kratos_values_override, var.kratos_values_override]
}
```

Ory Integration

Installed with the Helm Terraform provider

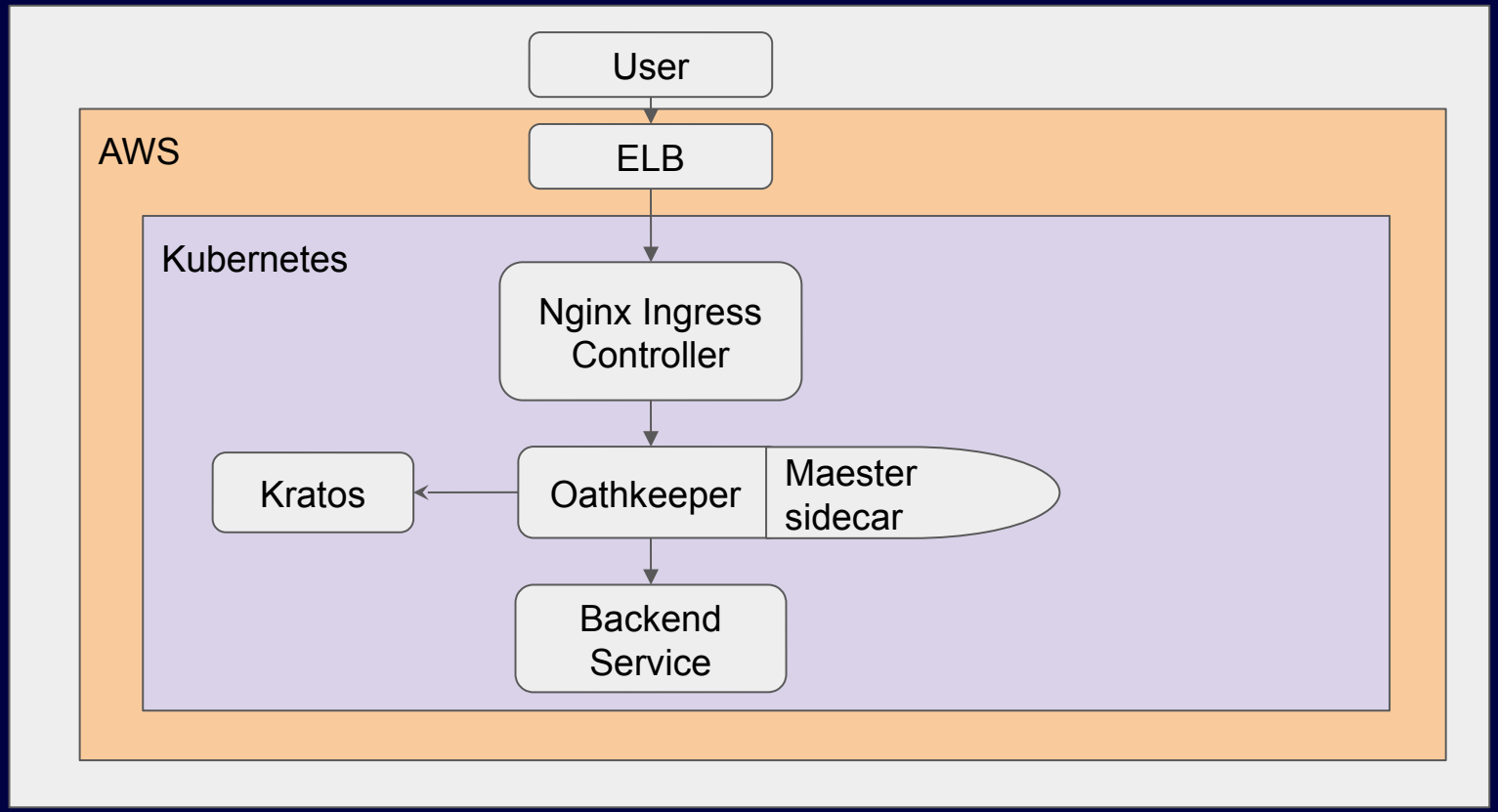
```
resource "helm_release" "kratos" {

  name       = "kratos-${var.name}"
  repository = "https://k8s.ory.sh/helm/charts"
  chart      = "kratos"
  version    = "0.4.11"
  namespace  = var.auth_namespace
  depends_on = [kubernetes_namespace.user_auth]

  values = [
    jsonencode(module.kratos_config.map_configs)
  ]
}
```

Ory Integration

With routes controlled at deploy-time by the application

```yaml
apiVersion: oathkeeper.ory.sh/v1alpha1
kind: Rule
metadata:
  name: authenticated-backend-endpoints
spec:
  upstream:
    preserveHost: true
    url: http://backend-service.project-namespace
  match:
    url: http://api.my-domain.com/<(?!(status|webhook|\.ory\/kratos)).*>
    methods:
      - GET
      - POST
  authenticators:
    - handler: cookie_session
  authorizer:
    handler: allow
  mutators:
    - handler: id_token
    - handler: header
```

Ory Integration



Ory Integration

From the application PoV

```
const authMiddleware = (req, res, next) => {
    const hasUserData = req.headers["x-user-id"] && req.headers["x-user-email"];
    if (!hasUserData) {
        res.status(401);
        res.json({
            success: false,
            message: "unauthenticated",
        });
    } else {
        req.user = {
            id: req.headers["x-user-id"],
            email: req.headers["x-user-email"],
        };
        next();
    }
};
```

Links

getzero.dev

Documentation - getzero.dev/docs

slack.getzero.dev

Case studies

- “Starting with Zero”
- “How I built a mini PaaS with Zero”