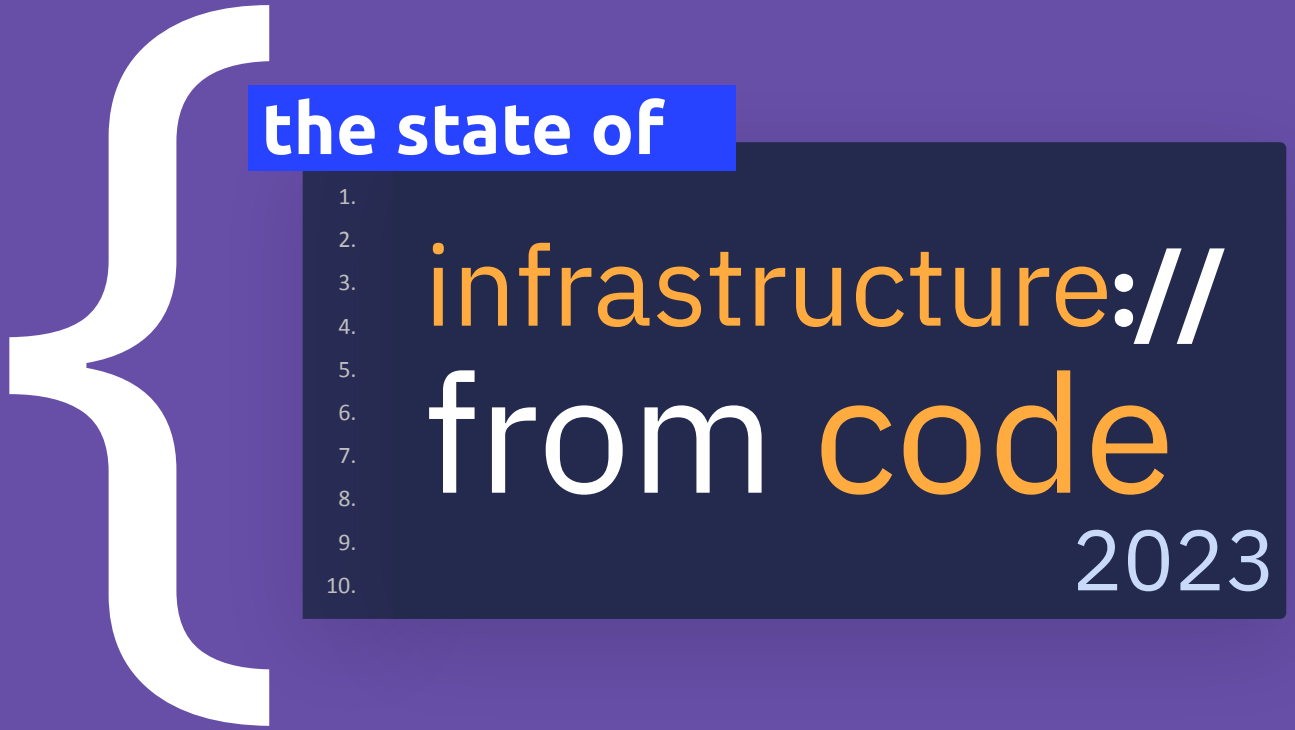

the state of
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
infrastructure://
from code
2023

Ala Shiban

Klotho

klo.dev/ifc

Microsoft

RIOT GAMES

> **talk.go**

1. What is Infra-**from**-code (IfC)
2. Comparison with Infra-as-code
3. Approaches in 2023
4. //TODO: Ping the infra team

> **what-is-infra-from-code.ts**

Tools that **infer** the cloud resources needed to deploy and run backend application **code**

```
//TODO: Show why it's not a great name
```

> what-is-infra-from-code.ts

> **names.txt**

What’s in a name?

> **comparison-with-iac.py3**

```
Infrastructure
as code

//NOTE: you usually write this
```

- Terraform
- Pulumi
- Cloudformation
- CDK

> **comparison-with-iac.py3**

Infrastructure (representation)
as code (or yaml)

//NOTE: you usually write this

- Terraform
- Pulumi
- Cloudformation
- CDK

example.tf

```
# Create a VPC and subnet for ElastiCache
resource "aws_vpc" "example_vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "example_subnet" {
  vpc_id     = aws_vpc.example_vpc.id
  cidr_block = "10.0.1.0/24"
}

# Create the ElastiCache instance
resource "aws_elasticache_cluster" "example_redis" {
  cluster_name = "example-redis-cluster"
  engine       = "redis"
  node_type    = "cache.t2.micro"
  num_cache_nodes = 1
  subnet_group_name = "example_redis_subnets"

  tags = {
    Name = "example-redis"
  }
}

# Create a Lambda function
resource "aws_lambda_function" "example_lambda" {
  function_name = "example-lambda"
  handler = "index.handler"
  runtime = "nodejs14.x"
  role = aws_iam_role.lambda_role.arn
  filename = "example_lambda.zip"
}
```

```
# Create an IAM role for the Lambda function
resource "aws_iam_role" "lambda_role" {
  name = "example-lambda-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "lambda.amazonaws.com"
        }
      }
    ]
  })
}

# Attach a policy to the IAM role
resource "aws_iam_role_policy_attachment" "lambda_policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonElastiCacheFullAccess"
  role = aws_iam_role.lambda_role.name
}

# Create an API Gateway
resource "aws_api_gateway_rest_api" "example_api" {
  name = "example-api"
}

# Create a Lambda integration for the API Gateway
resource "aws_api_gateway_integration" "example_integration" {
  rest_api_id = aws_api_gateway_rest_api.example_api.id
  resource_id = aws_api_gateway_resource.example_resource.id
  http_method = aws_api_gateway_method.example_method.http_method
  integration_http_method = "POST"
  type = "AWS_PROXY"
  uri = aws_lambda_function.example_lambda.invoke_arn
}
```

```
# Create a method for the API Gateway
resource "aws_api_gateway_method" "example_method" {
  rest_api_id = aws_api_gateway_rest_api.example_api.id
  resource_id = aws_api_gateway_resource.example_resource.id
  http_method = "POST"
}

# Add permission for the API Gateway to invoke the Lambda function
resource "aws_lambda_permission" "example_permission" {
  statement_id  = "AllowAPIGatewayInvoke"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.example_lambda.function_name
  principal     = "apigateway.amazonaws.com"

  source_arn = "${aws_api_gateway_rest_api.example_api.execution_arn}/*/*"
}
```

```
# Add the Redis endpoint as an environment variable for the Lambda function
resource "aws_lambda_function_environment" "example_lambda_env" {
  function_name = aws
}

# Create a resource for the API Gateway
resource "aws_api_gateway_resource" "example_resource" {
  rest_api_id = aws_api_gateway_rest_api.example_api.id
  parent_id   = aws_api_gateway_rest_api.example_api.root_resource_id
  path_part   = "example"
}
```

> **infra-from-code-approach.py3**

```
Infrastructure
from code
```

> **infra-from-code-approach.py3**

Infrastructure (Inferred)
from (Application) code

```
//TODO: Come up with a better name
```

> **example.py**

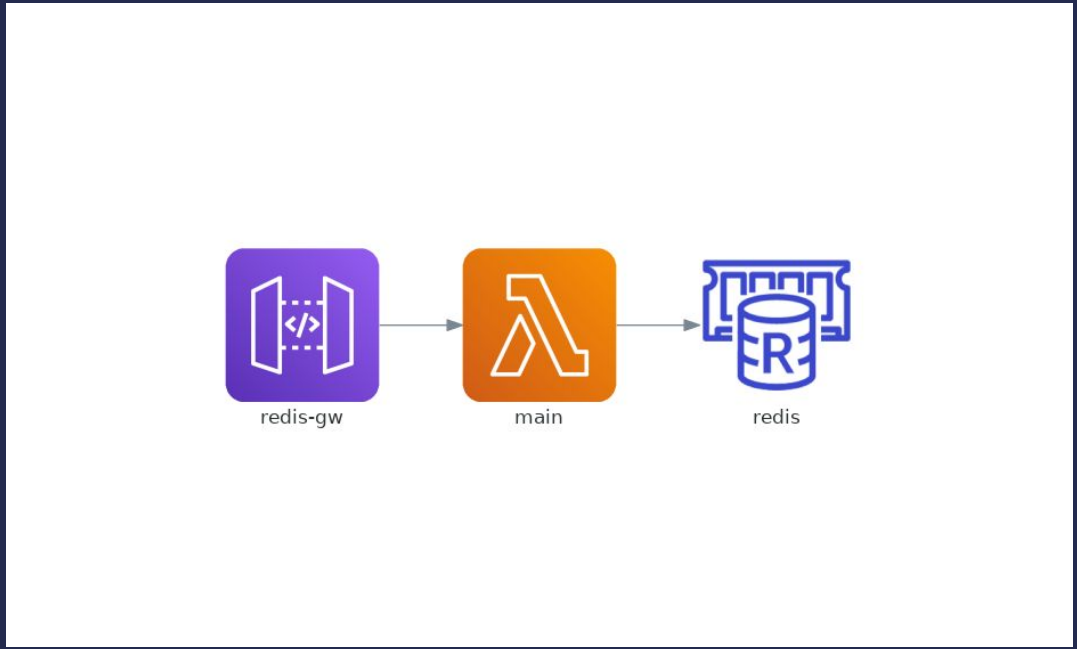
```python
from fastapi import FastAPI
import redis

# @klotho::persist {
#   id = "redis"
# }
client = redis.Redis(host='localhost', port=6379, db=0)

# @klotho::expose {
#   id = "redis-gw"
#   target = "public"
# }
app = FastAPI()

@app.get("/users/{name}", response_class=PlainTextResponse)
async def get_user(name: str):
    user = client.get(name)
    return user
```

> Example



> aspiration.md

Universal backend code

> what-problems-does-it-solve.cs

How does that promise go?

> **what-problems-does-it-solve.cs**

How does that promise go?

“Just focus on the ...”

> what-problems-does-it-solve.cs

> what-problems-does-it-solve.cs

* also pls learn istio, etcd, gRPC, cold boots, IAM, queues, terraform, docker, secret stores, distributed tracing, auto-scaling rules. You have until Thursday.

>

what-problems-does-it-solve.cs

1. Make our app aware of Lambda invoke

2. Create new IAC to provision the API Gateway and wire it to the Lambda process

3. Setup new IAM policies and security groups

4. Packaging our app in a docker container

5. Uploading it to ECR

6. Setting up a deploy pipeline to set up the infrastructure and configure the Lambda to use our container.

7. IAC to create a DynamoDB

8. Picking an SDK to interface with DynamoDB

9. Rewriting our app to create schemas, index, and make use of DynamoDB

10. Breaking our app into multiple repos or setting up hooks for a monorepo

11. Creating dockerfiles for all 3 new services

12. Setting up discovery, configuration, and more for each service

13. Rewriting all the IAC for the more complex topology

14. There are so many things you didn't even notice I'm here

15. IaC for SNS

16. Setting up topics and rewriting the application to use them

17. Doing work to determine how wakeup with SNS would work with Lambdas and testing it.

18. More IAM rules

> **what-problems-does-it-solve.cs**

19. Spin up VPCs

20. Configure subnets, security groups, IAM permissions

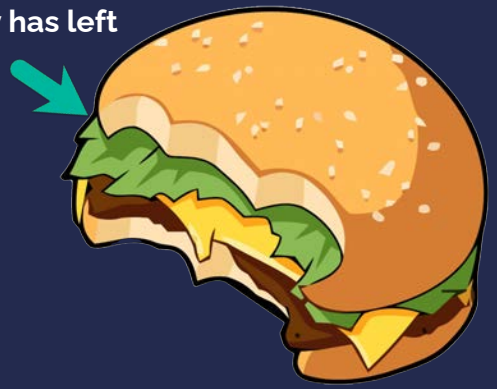
21. Redo your CI/CD pipeline to target your K8s cluster

22. Reorganize and rewrite your application to use the EKS interface (REST) vs Lambda (Invoke)

23. Revisit inter-module communication Create Helm charts, and other YAML specs for K8s

24. Spin up and configure an EKS cluster

What the company has left



Feature Dev Capacity: 63%

> **what-problems-does-it-solve.cs**

"Oh, can you…"

1. Run on AWS/GCP/Azure too?
2. Add fault tolerance / multi-region?
3. Switch from X to Kubernetes?
4. Integrate this serverless code?
5. Split that growing service into 2?
6. Combine those 3 services into 1?
7. Add gRPC for low latency

> what-problems-does-it-solve.cs

Infrastructure-from-Code aspires to enable Universal Backend Code

> what-problems-does-it-solve.cs

Enables developers to **only** write application code

> what-problems-does-it-solve.cs

Enables infra-platform teams to **set** tech standards and **ensure** feature teams adhere to them.

> what-problems-does-it-solve.cs

Together, with autonomy

> inference-approaches.tf

1. SDK Based
2. Language Based
3. Annotations Based
4. Framework Based

> **sdk-approach.go**

```
import { api } from '@nitric/sdk';

const helloApi = api('main');

helloApi.get('/hello/:name', async (ctx) => {
    const { name } = ctx.req.params;
    ctx.res.body = `Hello ${name}`;
});
```

> **sdk-approach.go**

Create collections

```
import { collection } from '@nitric/sdk';

const countries = collection('Countries').for('reading', 'writing', 'deleting');
```

Write documents

```
await countries.doc('USA').set({
  name: 'United States of America',
  population: 329500000,
});
```

Read documents

```
const doc = await countries.doc('USA').get();
```

> language-approach.go

```
bring cloud;

let bucket = new cloud.Bucket();

new cloud.Function(inflight (_: str): str => {
  bucket.put("hello.txt", "world");
});
```

> annotations-approach.go

TS hello-sender.ts

```ts
import "./reciever"
myEmitter.emit("hello", "world")
```

TS reciever.ts

```ts
/* @klotho::pubsub {
*  id = "emitterExample"
* }
*/
export const myEmitter = new events.EventEmitter()

myEmitter.on('hello', async (name) => {
    console.log(`${name} Event Received!`)
})
```

> hybrid-approaches.go

```rust
use rocket::{get, routes, State};
use sqlx::PgPool;

struct MyState(PgPool);

#[get("/hello")]
fn hello(state: &State<MyState>) -> &'static str {
    // Do things with `state.0`...
    "Hello, Postgres!"
}

#[shuttle_service::main]
async fn rocket(
    #[shared::Postgres] pool: PgPool
) -> shuttle_service::ShuttleRocket {
    let state = MyState(pool);

    Ok(
        rocket::build()
            .manage(state)
            .mount("/", routes![hello])
    )
}
```

> 2023.md

The Key Players 2023

[LOGOS]

> comparison.txt

Comparison Dimensions

License	Interface	Adaptive
Security	IaC	Backing Tech
Hosting Model	Portable	Config

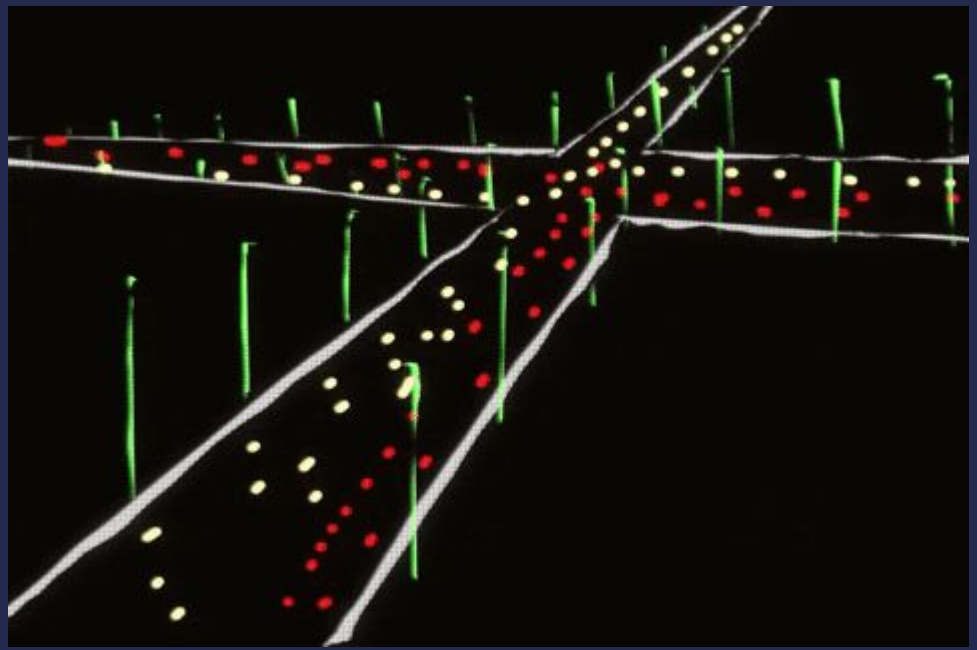
> LICENSE.TXT

> dev-interface.txt

> language.txt

> Security

> IaC Support



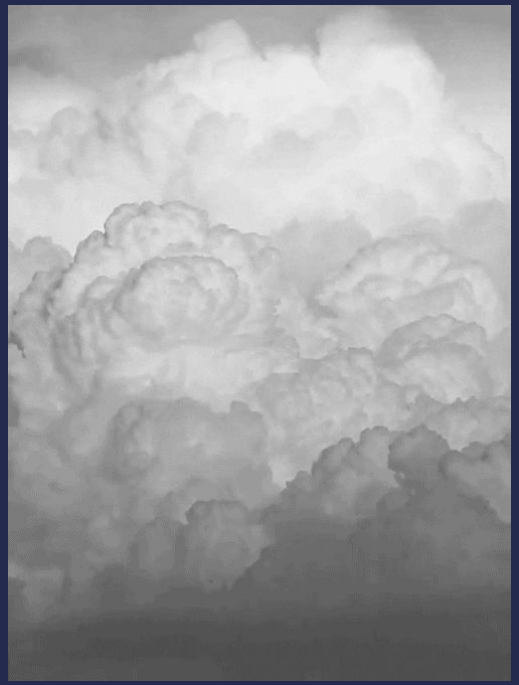
> Low Level Configurability

Hosting Model

Compute Backing Technologies

> Adaptability

Cloud Portability



The Matrix

A	B	C	D	E	F	G	H	I	J	K	L	M
	Language Support	Technologies Used	Compute Backing Tech	Hosting Model	License	Developer	Security	Low Level Config	Out-of-the-box Cloud Support	IaC Support	Project Age	Github Star Count (If Applicable)
Service Weaver (Google)	Go	Pre-selected	Kubernetes	Self-Hosted	Apache License 2.0	SDK	N/A		Google Cloud	✅	4 months	3000
Caios	Python	Adaptive	Serverless Kubernetes	Self-hosted	TBD	Standard/OSS interfaces	Least Privilage	High-level	AWS	✅	3 years	N/A
Modal	Python	Proprietary	Proprietary	PaaS	Closed Source	Annotations	N/A		None	❌	2 Years	100
Klotho	JS TS Go Python C#	Adaptive	Serverless Kubernetes	Self-Hosted	Apache 2.0	Standard/OSS Frameworks + Annotations	Least Privilage	High Level + Low Level using IaC	AWS	✅	3 years	610
Encore	Go	Pre-selected	Serverless	PaaS (Full Featured) Self-Hosted (Limited)	Mozilla Public License 2.0	SDK + Annotations	Least Privilage		AWS, GCP, Azure	❌	2 years	3600
Nitric	JS TS Go Python Java C#	Adaptive	Serverless	PaaS Self-Hosted	Apache License 2.0	SDK	Least Privilage		AWS, GCP, Azure	✅	2.5 Years	498
Chalice	Python	Explicit	Serverless	Self-Hosted	Apache License 2.0	Annotations	Least Privilage		AWS	❌	7 Years	9600
Wing	Wing NPM/JS/TS interop	Adaptive	Serverless	Self-Hosted	MIT	Language	Least Privilage	High-level + Config plugins	AWS, Azure, GCP	✅	1 year	282
Shuttle	Rust	Pre-selected	Serverless	PaaS	Apache License 2.0	SDK + Annotations		N/A	AWS	❌	1 Year	2000

klo.dev/ifc

> Infra-Platform-Teams

> **SIGTERM**

Break free from the
cycle of the mundane
to work on challenges
no one else is, and
only you can.