BREAKING THE PRIME DIRECTIVE

USING C# FOR IAC TO PUT .NET ON AWS

Daniel Ward

THE PRIME DIRECTIVE

1. No identification of self or mission.
2. No interference with the social, cultural, or technological development of said planet.
3. No reference to space, other worlds, or advanced civilizations.

THE PRIME DIRECTIVE V2

.NET

OVERVIEW

- Why .NET on AWS?
- Tooling
- .NET on AWS Community
- Native AOT
- AWS Cloud Development Kit (CDK)
- Coding demo
- Won’t cover
  - Creating an AWS account and IAM user
  - Installing command line tools

ME

- Software consultant LEAN TECHNIQUES leantechniques.com
- Co-organizer of the San Antonio/Austin .NET User Group
- daninacan.com
- @danielwarddev
- daniel-ward-dev

SAN ANTONIO .NET

AUSTIN .NET

WHY .NET ON AWS?

- ⭐.NET and AWS is what you know!
- AWS supports Windows environments
- AWS supports .NET AoT on Lambda
- .NET tooling for AWS

.NET ON AWS TOOLING

- Lambda runtimes
- IDE/CLI Lambda templates and tools - https://docs.aws.amazon.com/lambda/latest/dg/csharp-package-cli.html
- .NET Lambda annotations - https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/aws-lambda-annotations.html

.NET ON AWS COMMUNITY/RESOURCES

- https://aws.amazon.com/developer/language/net/net-community/
- .NET on AWS Show – https://community.aws/@codetraveler
- .NET on AWS twitter account - @dotnetonAWS
- AWS + .NET Foundation – https://aws.amazon.com/blogs/opensource/aws-joins-the-net-foundation/
- Serverless Land – https://serverlessland.com/lambda/dotnet
- AWS blog posts

NATIVE AOT (.NET 7+)

- Just-in-time (JIT) process:
  1. Compiles your written code to common intermediate language (CIL)
  2. Your CIL is shipped and is compiled to native code at execution time with a JIT compiler
  3. Native code is ran
- Native AOT:
  1. Compiles your written code to CIL
  2. A native AOT compiler compiles the CIL to native code at publishing time by targeting a specific environment (Linux x64, Windows x64, etc.)

NATIVE AOT ON AWS

- Up to 86% improvement in Lambda cold start times
- https://aws.amazon.com/blogs/compute/introducing-the-net-8-runtime-for-aws-lambda/
- https://docs.aws.amazon.com/lambda/latest/dg/dotnet-native-aot.html
- https://blog.martincostello.com/native-aot-make-dotnet-lambda-go-brr/

AWS CLOUD DEVELOPMENT KIT (CDK)

- https://docs.aws.amazon.com/cdk/v2/guide/cli.html
- Use a programming language to create CloudFormation (IaC)
  - C#, Java, Javascript/Typescript, Python, Go
- Benefits of IaC while also getting the benefits of a programming language
  - Familiarity
  - Intellisense
  - Type safety

CDK CONSTRUCTS

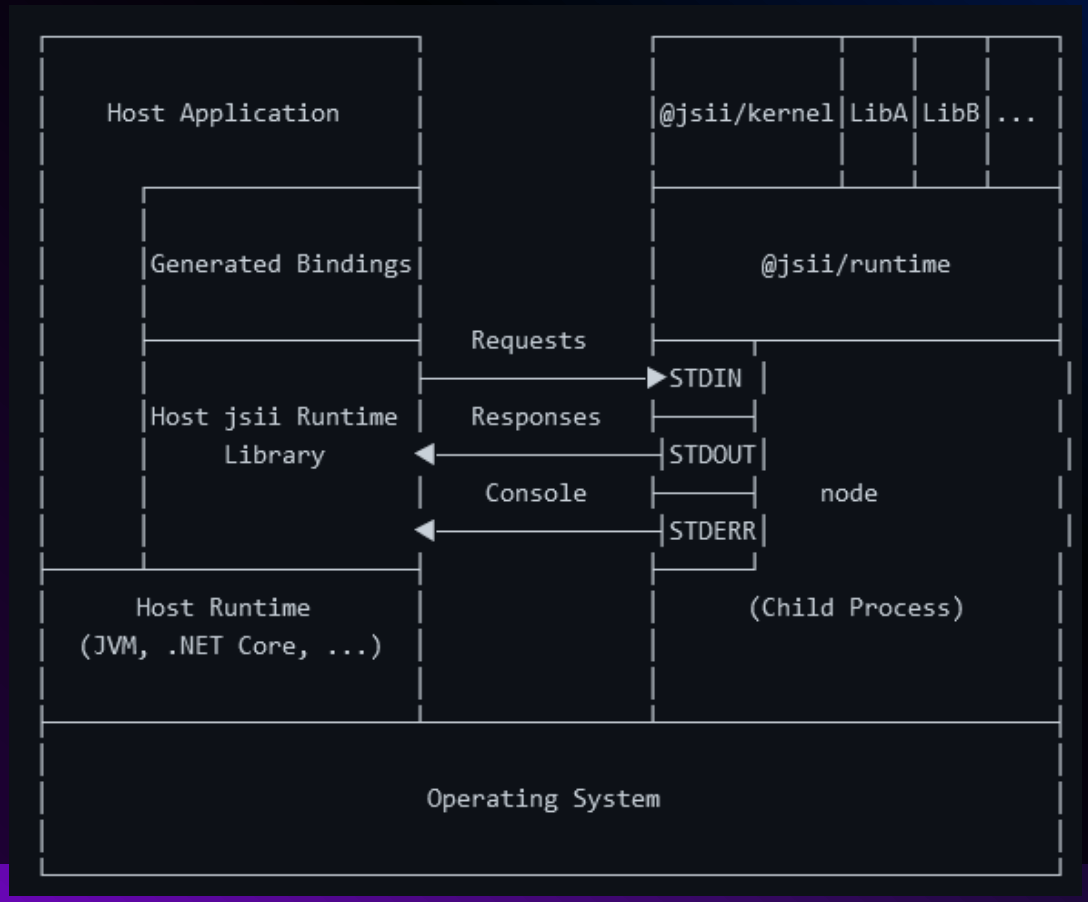
- Represents one or more CFN resources
- Sensible defaults
- Common pattern
- Best practices
- L1 (aka CFN resources) – Example: CfnBucket
- L2 (aka curated constructs). Example: Bucket
- L3 (aka patterns). Example: ApplicationLoadBalancedFargateService

REQUIREMENTS

- .NET SDK
- Docker
- An AWS account
- An IAM user
- AWS CLI
- AWS CDK
- Node.js (I recommend that you install nvm first and use that to install Node)

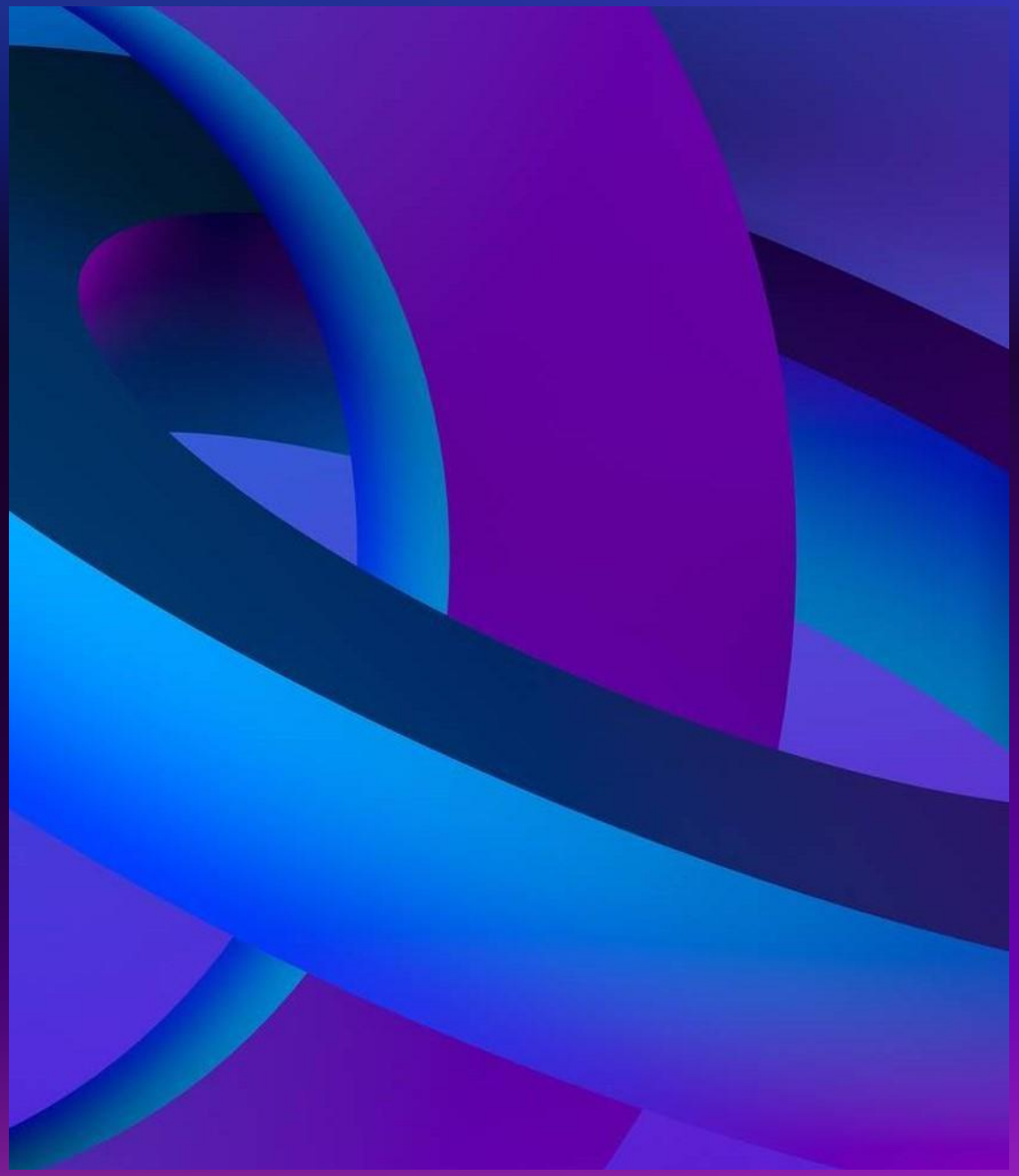
WAIT! WHY DO I NEED NODE?

- CDK is made with jsii
- jsii exposes a Typescript API and generates code in the specified language to model that same API
- The generated code still calls down to Javascript code through a sidekick node process, which responds back with the result
- https://aws.github.io/jsii/overview/runtime-architecture/



CODING

DEMO



CATTLE, NOT PETS

- Bill Baker, 2011/2012. Scale-up vs scale-out
- https://cloudscaling.com/blog/cloud-computing/the-history-of-pets-vs-cattle/

Pets

- Irreplaceable and cute
- Individual names
- Manually managed

Cattle

- Replaceable and fine
- Numbered
- Automated management

RECAP

- Why .NET on AWS?
- Tooling
- .NET on AWS Community
- Native AOT
- AWS Cloud Development Kit (CDK)
- Cattle vs pets
- Using the CDK
- C# Lambda
- Lambda annotations for DI
- DynamoDB
- API Gateway

THANK YOU

- daninacan.com
- @danielwarddev
- daniel-ward-dev