CloudFormation Custom Resource Providers

Beyond AWS Native Resource Types

by Ruben de Gooijer

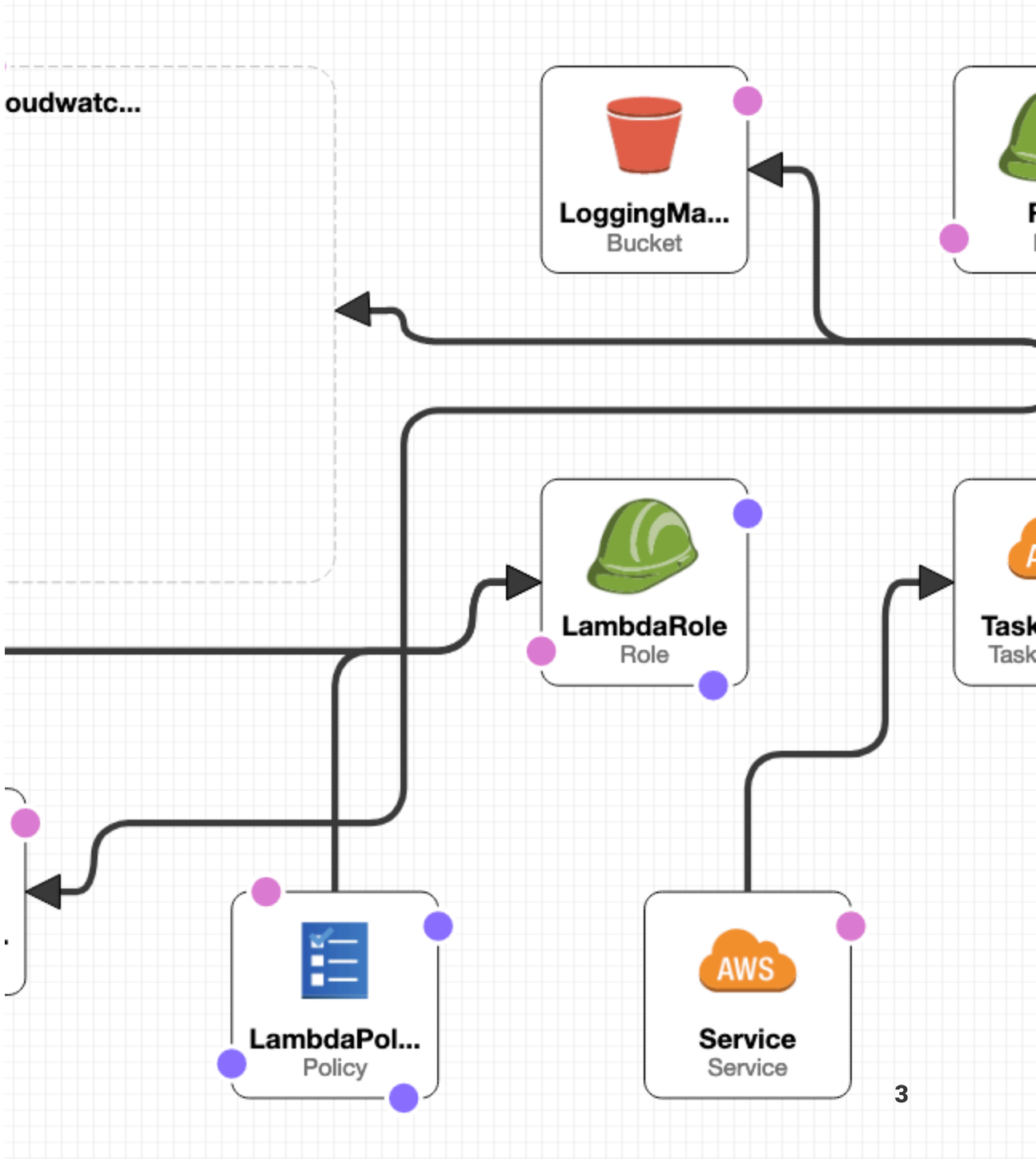
Agenda

1. Problem
2. Solution
3. Addition Reinvented
4. Conclusion
5. Q&A

Infrastructure As Code

Declare the desired state of your infrastructure, and CloudFormation does the rest:

- Change detection
- Reconcile current state with desired state
- Dependency management
- Handle intermittent failures
- Throttling of API calls
- ...



What about non-AWS resources?

- **Third-parties**
  - DataDog, Selenium, ...
- **Internal APIs**
  - Admin user, service configuration, ...
- **Features not supported by native resource types**
  - More advanced secret generation, helper functions, ...
- **Use of external data sources**
  - Readonly configuration hosted elsewhere, ...

Mechanisms

There are two extension mechanisms for CloudFormation:

- **Custom Resource Providers**
- **Resource Types**

The key difference is the level of integration with CloudFormation.

Registry

Resource types | Modules

Resource types (667)

Discover the new AWS, third-party, and organization-specific resource types available for use in your stack templates. This includes any private resources unique to your a

Public ▲

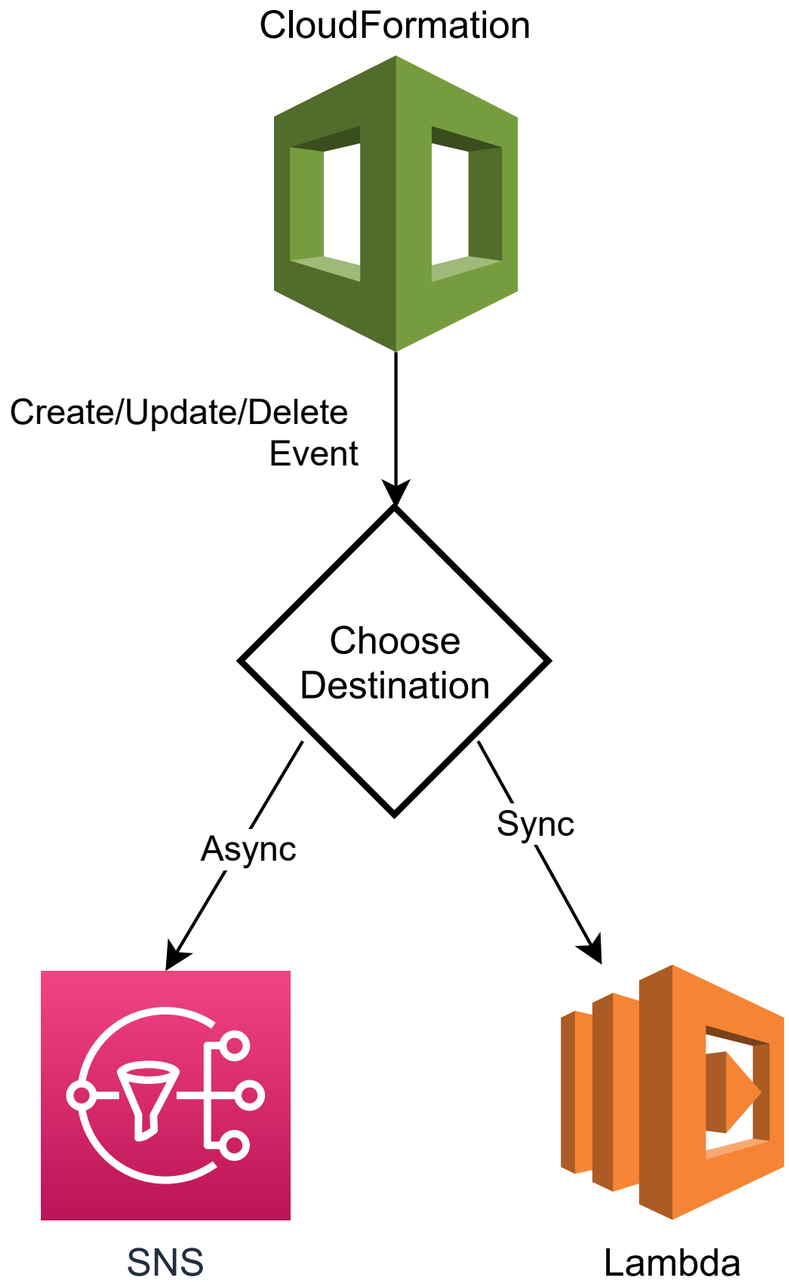
Public

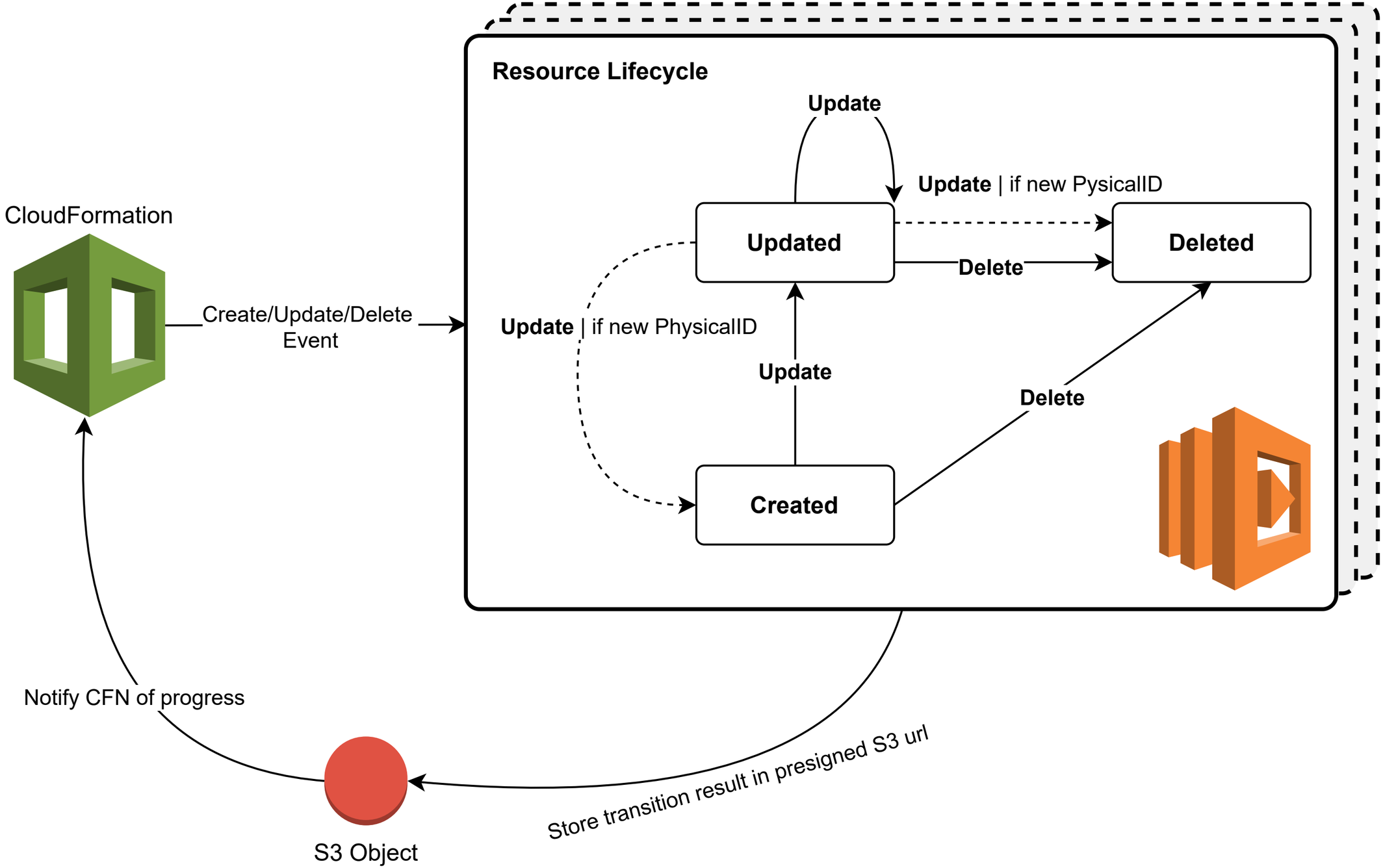
Private

Default version	Registry	Publisher	Description
AWS::WAFv2::RegexPatternSet	-	Public	AWS	Contains a list of Regular expressions based on
AWS::WAFv2::RuleGroup	-	Public	AWS	Contains the Rules that identify the requests th
AWS::WAFv2::WebACL	-	Public	AWS	Contains the Rules that identify the requests th
AWS::WAFv2::WebACLAssociation	-	Public	AWS	Associates WebACL to Application Load Balanc
AWS::WorkSpaces::ConnectionAlias	-	Public	AWS	Resource Type definition for AWS::WorkSpaces
AWS::WorkSpaces::Workspace	-	Public	AWS	
Alexa::ASK::Skill	-	Public	Alexa	

```
function AddFn(A, B) {
  return A + B;
}

AdditionResultResource
  = AddFn(2, 2);
```


CloudFormation
Create/Update/Delete
Event
Choose
Destination
Async
Sync
SNS
Lambda


Resource Lifecycle
Update
Update | if new PysicalID
Updated
Deleted
Delete
Update | if new PhysicalID
Update
Delete
Created
CloudFormation
Create/Update/Delete
Event
Notify CFN of progress
Store transition result in presigned S3 url
S3 Object

```yaml
AddFn:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.lambda_handler
      Runtime: python3.8
      Code:
        ZipFile: | See right

AdditionResultResource:
    Type: Custom::Add
    Properties:
      ServiceToken: !GetAtt "AddFn.Arn"
      A: 2
      B: 2

Outputs:
  AdditionResult:
    Description: "Result of Addition"
    Value: !GetAtt AdditionResultResource.Result
```

```python
import cfnresponse, uuid

def lambda_handler(event, context):
  response_data = {}
  if event["RequestType"] == "Create":
    a = int(event["ResourceProperties"]["A"])
    b = int(event["ResourceProperties"]["B"])
    physicalResourceId = uuid.uuid4()
    response_data = { "Result": a + b }

  else: # "Update" or "Delete":
    physicalResourceId = event["PhysicalResourceId"]

  cfnresponse.send(
    event,
    context,
    cfnresponse.SUCCESS,
    response_data,
    physicalResourceId
  )
```

Idempotency

Create handler **MUST** be idempotent:

```
create(data) = create(create(data)) = create(create(create(data)))
```

Otherwise it will leak resources in the non-happy case
when resource creation is retried.

Stabilization Properties

- **Desired-state stabilization**
  - After a Create/Update, subsequent calls **MUST** be able to **read** the properties
- **Runtime-state stabilization**
  - After a Create/Update, subsequent calls **SHOULD** be able to **use** the resource

For simple resources stabilization can be immediate.

Not *always* the case - for example, when creating EC2 instances.

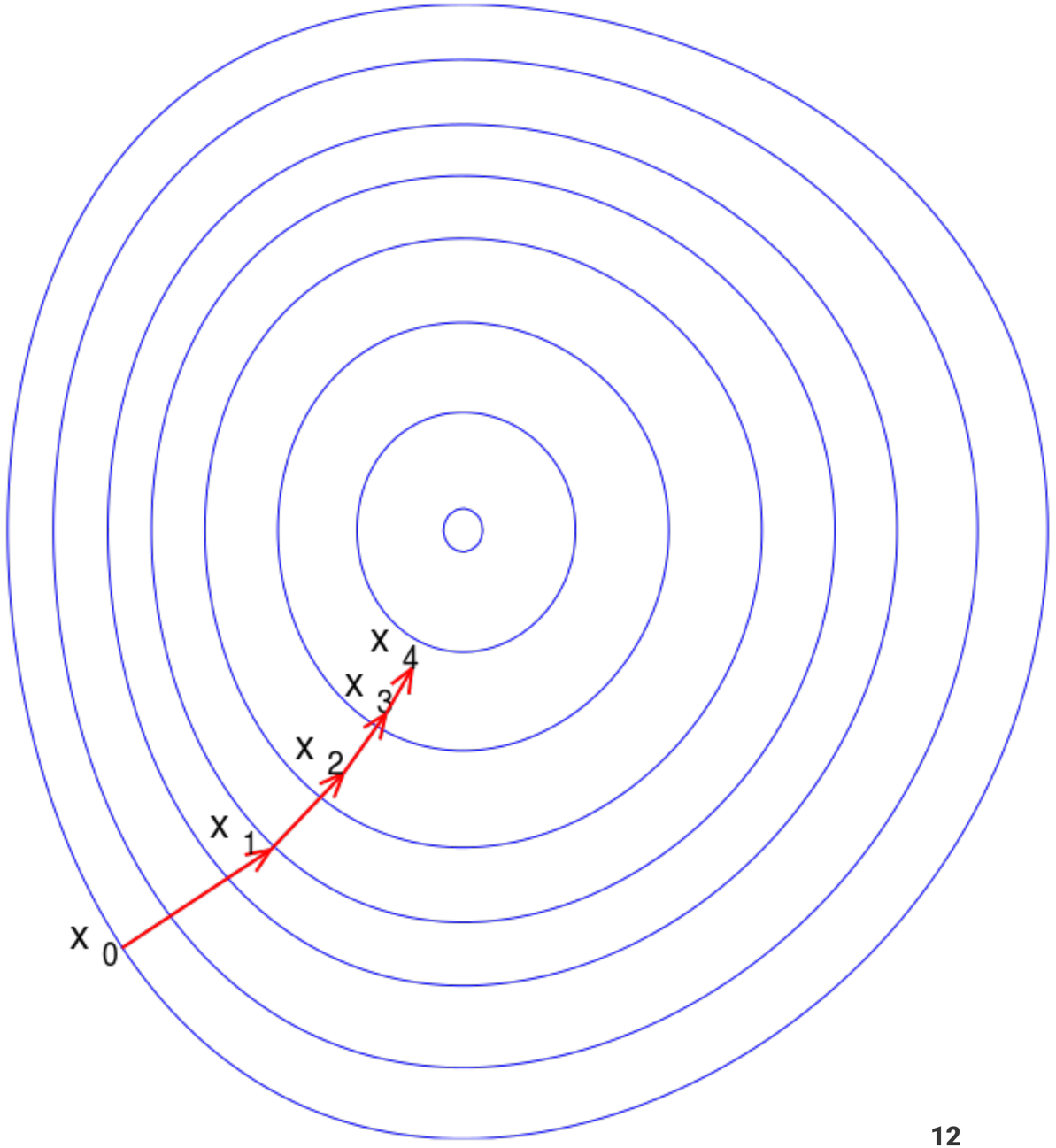


Image: https://en.wikipedia.org/wiki/Gradient_descent

Frameworks / Libraries

Ease implementation with helper library for handling common concerns.

	Binx.io cfn-resource-provider	**AWS custom-resource-helper**
Schema validation	Yes (json schema)	-
Error handling	Yes	Yes
Lifecycle hooks	Yes	Yes
Language	Python	Python (typed)
Logging	Basic	Basic + Context
Long running provisioning (+15min)	-	Yes (CloudWatch Events)

Custom Resource Provider or Resource Type

	Custom Resource Provider	**Resource Type**
Schema	Internal	External
Execution	Own Account	AWS Managed
Pricing	Underlying Resources	Per handler op*
Namespace	`Custom::*`	All Non-Reserved
Testing Framework	No	Yes
Conformance Testing	-	Yes
Language Support	Any	Java, Go, Python
Distribution	Copy Template	CFN Registry (public/private)

*** https://aws.amazon.com/cloudformation/pricing/**

What Others Are Doing

- Terraform custom providers
- Kubernetes custom resources
- Azure custom resource provider

Conclusion

- Quite easy to write; even easier with helper framework / library
- Be careful with secrets, use `NoEcho` option
- Cost/benefit of explicitly modelling Custom Resource Provider API
- When are custom resources not a good fit?
- **Resource Types** the future?

Image by Robin Higgins