Lesson 4 AWS CLI and CloudFormation

Michael Yang

AWS Tools to call API

- Command-line interface (CLI)—Use the CLI to call the AWS API from your terminal.
- Software development kit (SDK)—SDKs, available for most programming languages, make it easy to call the AWS API from your programming language of choice.
- AWS CloudFormation—Templates are used to describe the state of the infrastructure. AWS CloudFormation translates these templates into API calls.

DevOps

- The DevOps movement aims to bring software development and operations together. This usually is accomplished in one of two ways:
  - Using mixed teams with members from both operations and development. Developers become responsible for operational tasks like being on call. Operators are involved from the beginning of the software development cycle, which helps make the software easier to operate.
  - Introducing a new role that closes the gap between developers and operators. This role communicates a lot with both developers and operators and cares about all topics that touch both worlds.

Automation

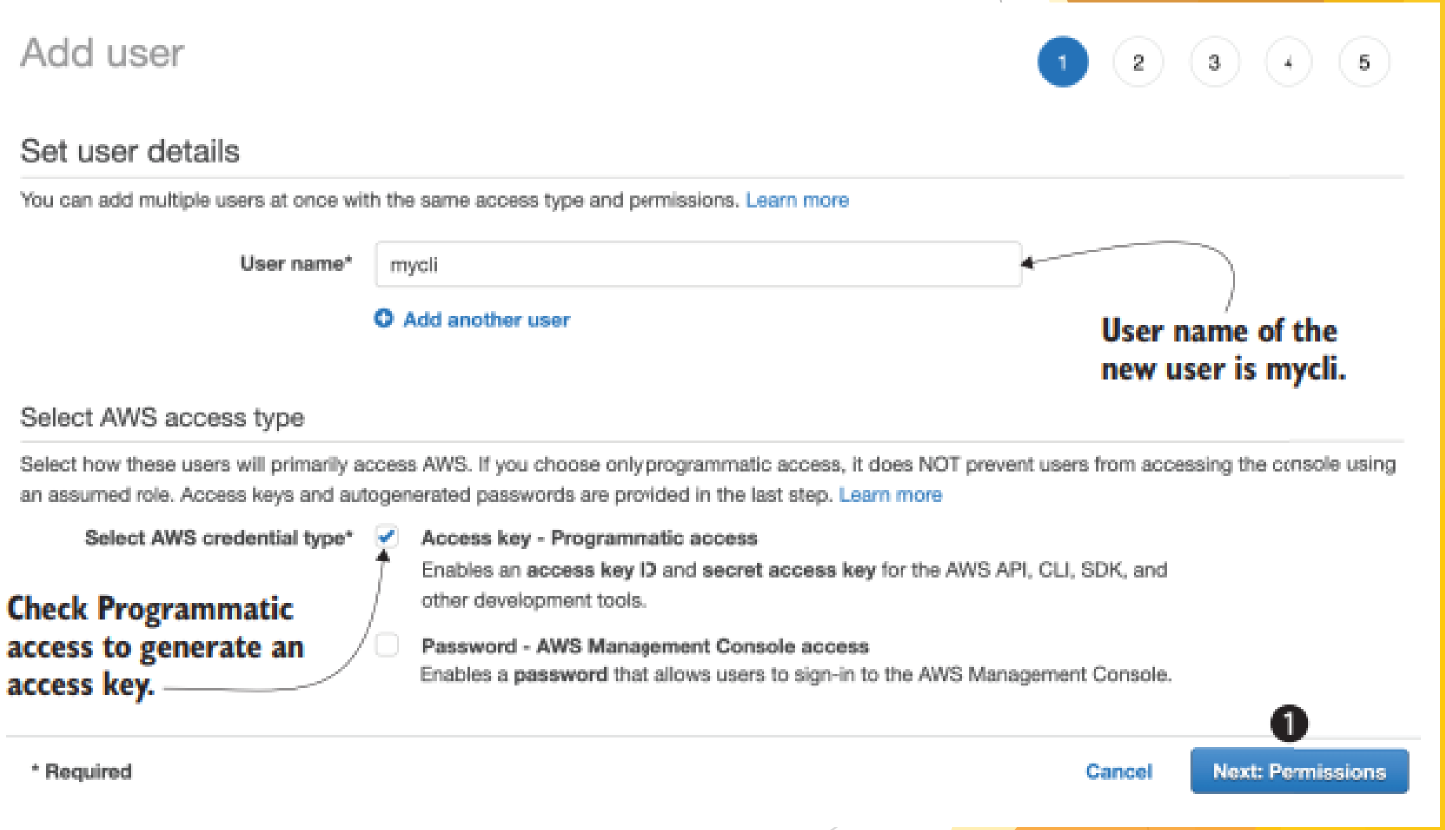
- Why should you automate instead of using the graphical AWS Management Console?
  A script or a blueprint can be reused and will save you time in the long run.
- Another benefit is that a script or blueprint is the most detailed documentation
  you can imagine (even a computer understands it).

Install AWS CLI

- The following steps guide you through installing the AWS CLI on Windows using the MSI Installer:
  1 Download the AWS CLI installer at https://awscli.amazonaws.com/AWSCLIV2.msi
  2 Run the downloaded installer, and install the CLI by going through the installation wizard.
  3 Run PowerShell as administrator by searching for “PowerShell” in the Start menu and choosing Run as Administrator from its context menu.
  4 Type `Set-ExecutionPolicy Unrestricted` into PowerShell, and press Enter to execute the command. This allows you to execute the unsigned PowerShell scripts from our examples.
- 5 Close the PowerShell window; you no longer need to work as administrator.
  6 Run PowerShell by choosing PowerShell from the Start menu.
  7 Verify whether the CLI is working by executing `aws --version` in PowerShell. The version should be at least 2.4.0.

Configuring CLI

- To create a new user, use the following steps:
  1 Open the AWS Management Console at https://console.aws.amazon.com.
  2 Click Services and search for IAM.
  3 Open the IAM service.
- 1 Click Add Users to open the page shown in figure 4.4.
  2 Enter mycli as the user name.
- 3 Under AWS credential type, select Access Key—Programmatic Access.
  4 Click the Next: Permissions button.



Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* mycli

Add another user

User name of the new user is mycli.

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*

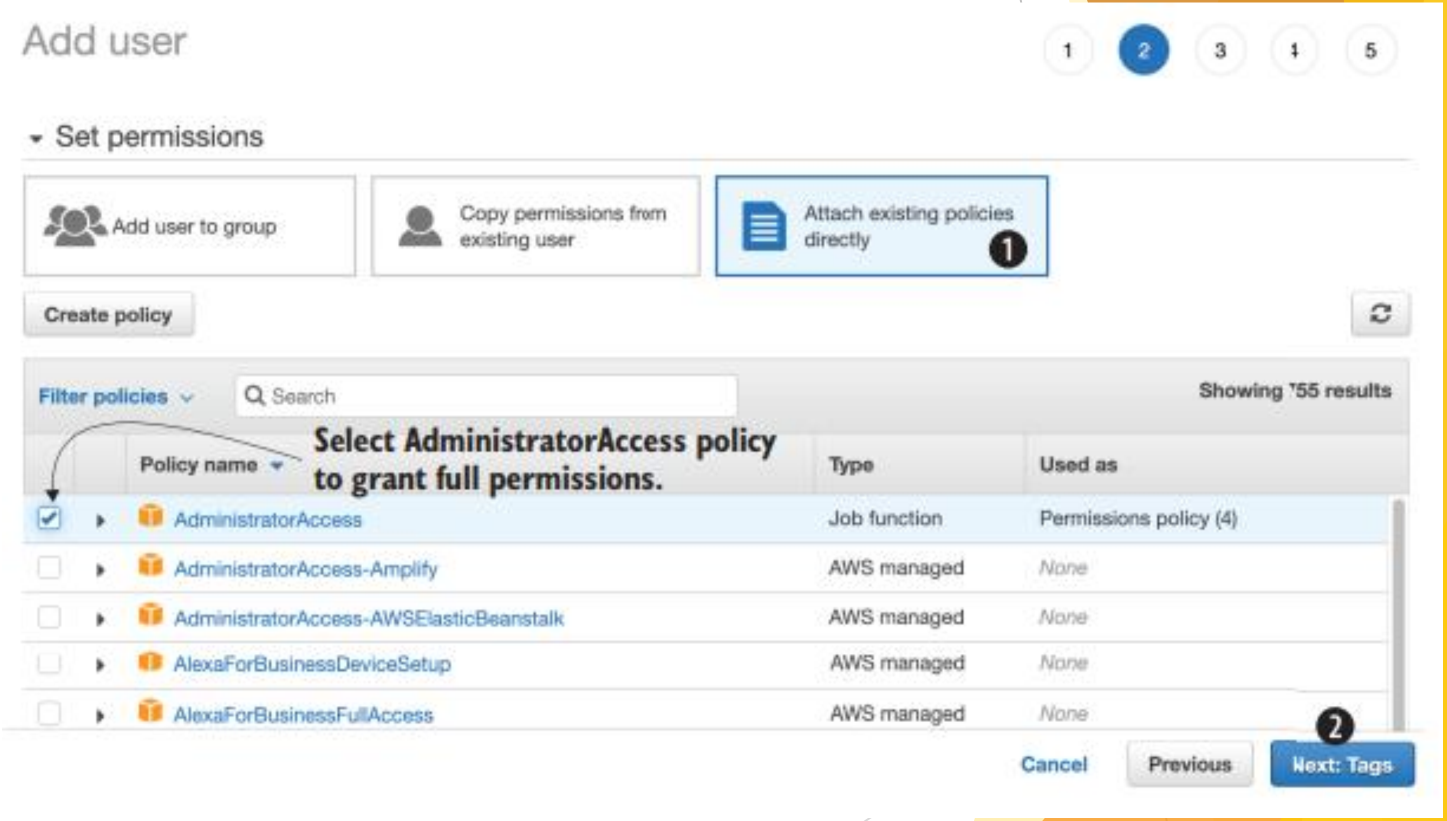
[✓] Access key - Programmatic access
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.

[ ] Password - AWS Management Console access
Enables a password that allows users to sign-in to the AWS Management Console.

Check Programmatic access to generate an access key.

Configuring CLI

- Define the permissions for the new user:
  1 Click Attach Existing Policies Directly.
  2 Select the AdministratorAccess policy.
  3 Click the Next: Tags button



Add user

1 2 3 4 5

Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly ❶

Create policy

Filter policies | Search | Showing 155 results

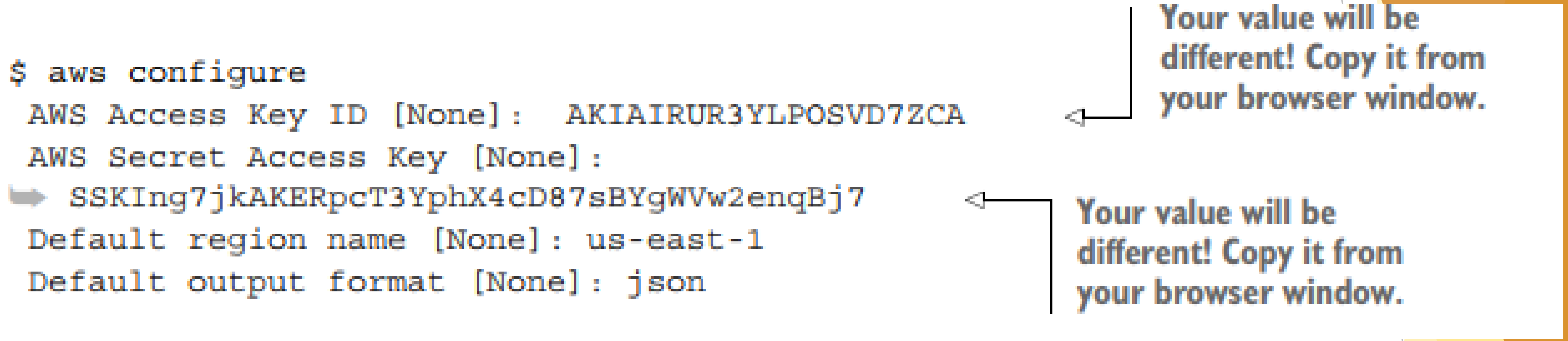
Select AdministratorAccess policy to grant full permissions.

	Policy name	Type	Used as
✓	AdministratorAccess	Job function	Permissions policy (4)
	AdministratorAccess-Amplify	AWS managed	None
	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None
	AlexaForBusinessDeviceSetup	AWS managed	None
	AlexaForBusinessFullAccess	AWS managed	None

❷

Cancel | Previous | Next: Tags

Configuring CLI



```
$ aws configure
 AWS Access Key ID [None]:  AKIAIRUR3YLPOSVD7ZCA
 AWS Secret Access Key [None]:
➥ SSKIng7jkAKERpcT3YphX4cD87sBYgWVw2enqBj7
 Default region name [None]: us-east-1
 Default output format [None]: json
```

Your value will be different! Copy it from your browser window.

Your value will be different! Copy it from your browser window.

Infrastructure as Code

- *Infrastructure as Code* is the idea of using a high-level programming language to control infrastructures.
- Infrastructure can be any AWS resource, like a network topology, a load balancer, a DNS entry, and so on.
- In software development, tools like automated tests, code repositories, and build servers increase the quality of software engineering.
- If your infrastructure is defined as code, then you can apply these types of software development tools to your infrastructure and improve its quality.

CloudFormation Template

- A basic CloudFormation template consists of the following five parts:

Format version—The latest template format version is 2010-09-09, and this is currently the only valid value. Specify this version; the default is to use the latest

version, which will cause problems if new versions are introduced in the future.

Description—What is this template about?

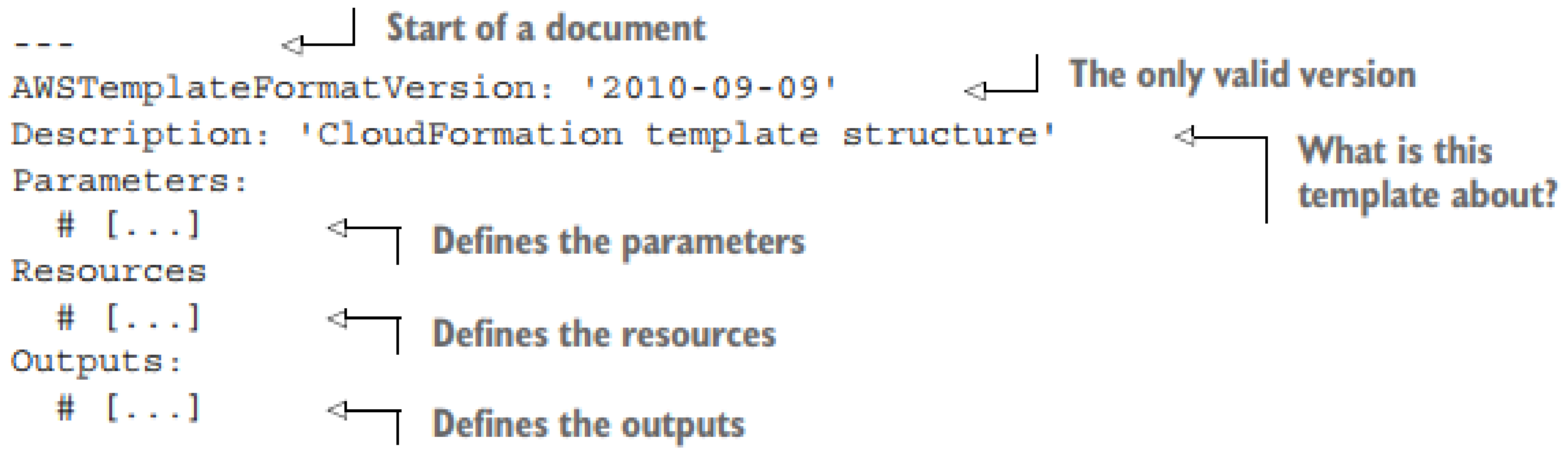
Parameters—Parameters are used to customize a template with values, for example, domain name, customer ID, and database password.

Resources—A resource is the smallest block you can describe. Examples are a virtual machine, a load balancer, or an Elastic IP address.

Outputs—An output is comparable to a parameter, but the other way around.

An output returns details about a resource created by the template, for example, the public name of an EC2 instance.

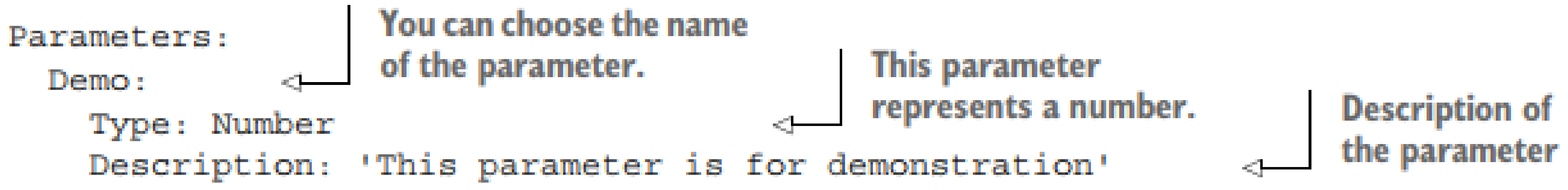
CloudFormation Template



```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'CloudFormation template structure'
Parameters:
  # [...]
Resources
  # [...]
Outputs:
  # [...]
```

Start of a document
The only valid version
What is this template about?
Defines the parameters
Defines the resources
Defines the outputs

CloudFormation Template



```
Parameters:
  Demo:
    Type: Number
    Description: 'This parameter is for demonstration'
```

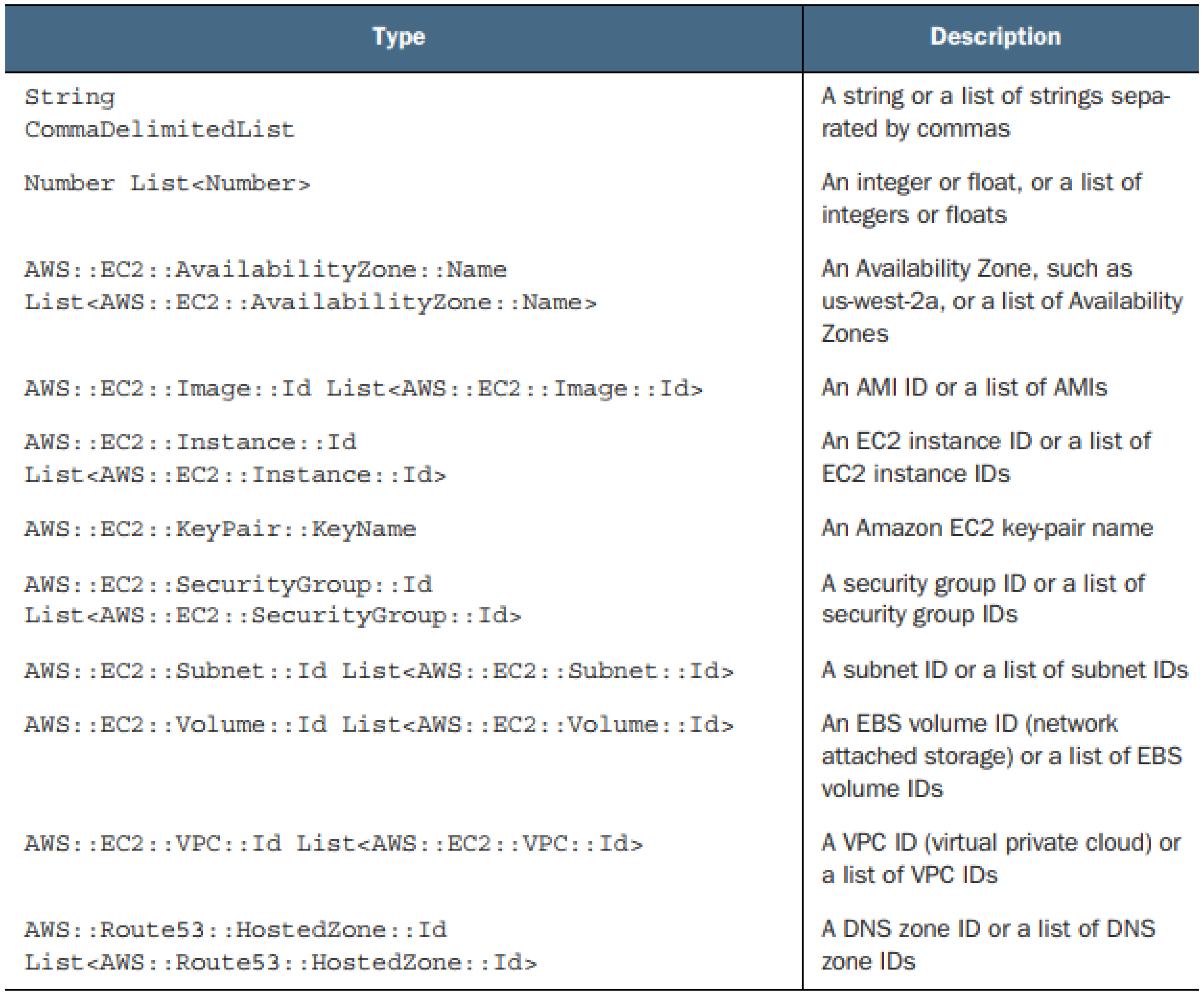
You can choose the name of the parameter.

This parameter represents a number.

Description of the parameter

CloudFormation Template

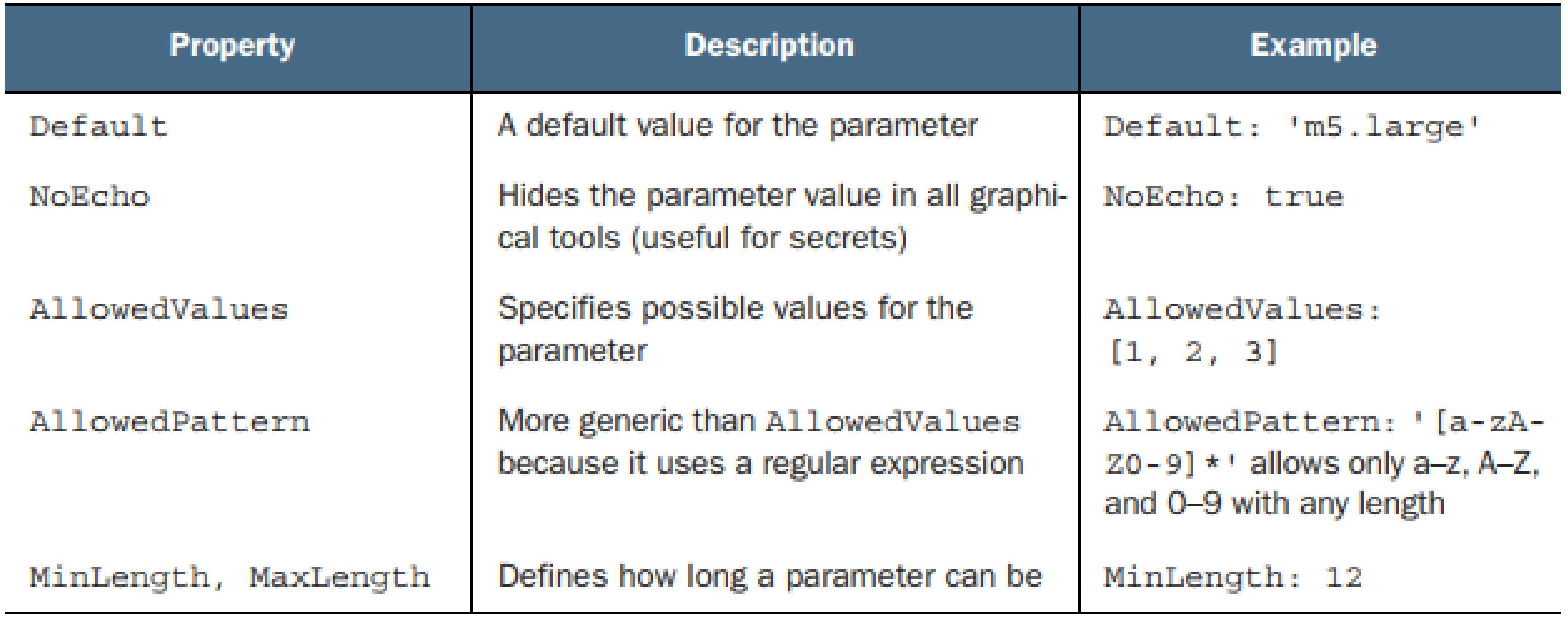
- Valid types are listed



Type	Description
`String` `CommaDelimitedList`	A string or a list of strings separated by commas
`Number List<Number>`	An integer or float, or a list of integers or floats
`AWS::EC2::AvailabilityZone::Name` `List<AWS::EC2::AvailabilityZone::Name>`	An Availability Zone, such as us-west-2a, or a list of Availability Zones
`AWS::EC2::Image::Id List<AWS::EC2::Image::Id>`	An AMI ID or a list of AMIs
`AWS::EC2::Instance::Id` `List<AWS::EC2::Instance::Id>`	An EC2 instance ID or a list of EC2 instance IDs
`AWS::EC2::KeyPair::KeyName`	An Amazon EC2 key-pair name
`AWS::EC2::SecurityGroup::Id` `List<AWS::EC2::SecurityGroup::Id>`	A security group ID or a list of security group IDs
`AWS::EC2::Subnet::Id List<AWS::EC2::Subnet::Id>`	A subnet ID or a list of subnet IDs
`AWS::EC2::Volume::Id List<AWS::EC2::Volume::Id>`	An EBS volume ID (network attached storage) or a list of EBS volume IDs
`AWS::EC2::VPC::Id List<AWS::EC2::VPC::Id>`	A VPC ID (virtual private cloud) or a list of VPC IDs
`AWS::Route53::HostedZone::Id` `List<AWS::Route53::HostedZone::Id>`	A DNS zone ID or a list of DNS zone IDs

CloudFormation Template

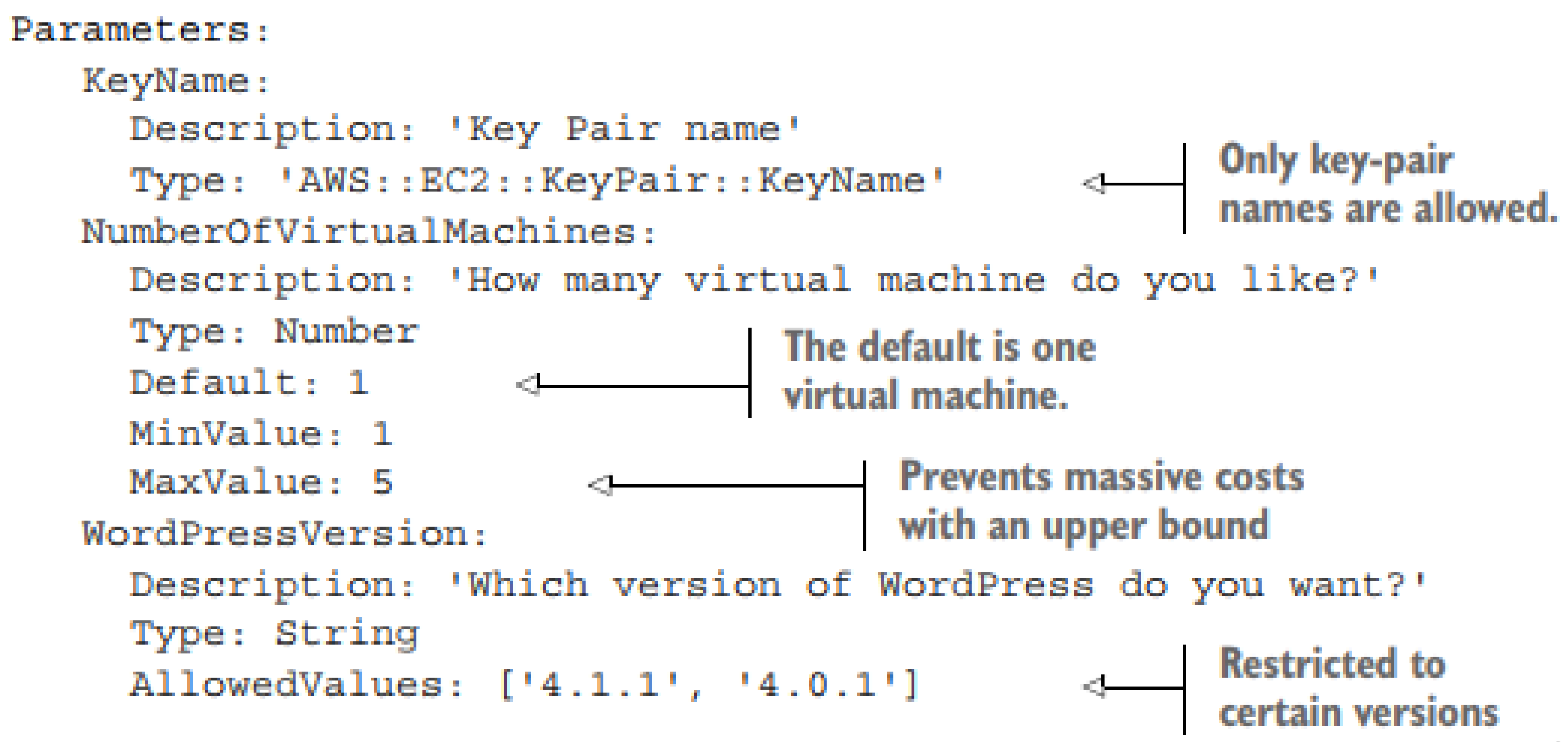
- In addition to using the `Type` and `Description` properties, you can enhance a parameter with the properties listed in table



Property	Description	Example
`Default`	A default value for the parameter	`Default: 'm5.large'`
`NoEcho`	Hides the parameter value in all graphical tools (useful for secrets)	`NoEcho: true`
`AllowedValues`	Specifies possible values for the parameter	`AllowedValues: [1, 2, 3]`
`AllowedPattern`	More generic than `AllowedValues` because it uses a regular expression	`AllowedPattern: '[a-zA-Z0-9]*'` allows only a–z, A–Z, and 0–9 with any length
`MinLength, MaxLength`	Defines how long a parameter can be	`MinLength: 12`

CloudFormation Template

A **parameter** section of a CloudFormation template could look like this



```
Parameters:
  KeyName:
    Description: 'Key Pair name'
    Type: 'AWS::EC2::KeyPair::KeyName'
  NumberOfVirtualMachines:
    Description: 'How many virtual machine do you like?'
    Type: Number
    Default: 1
    MinValue: 1
    MaxValue: 5
  WordPressVersion:
    Description: 'Which version of WordPress do you want?'
    Type: String
    AllowedValues: ['4.1.1', '4.0.1']
```

Only key-pair names are allowed.

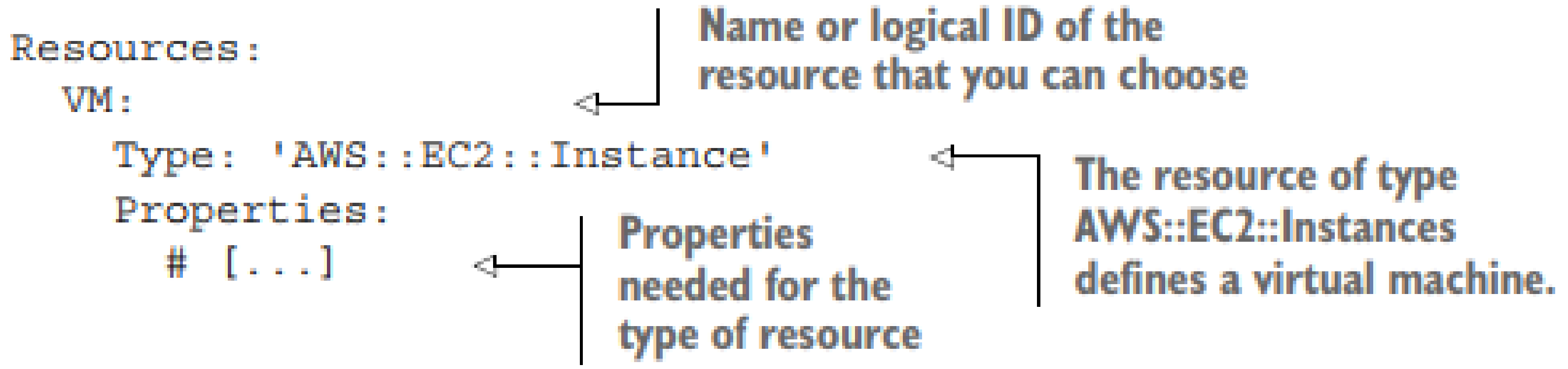
The default is one virtual machine.

Prevents massive costs with an upper bound

Restricted to certain versions

CloudFormation Template

- A **resource** has at least a name, a type, and some properties, as shown

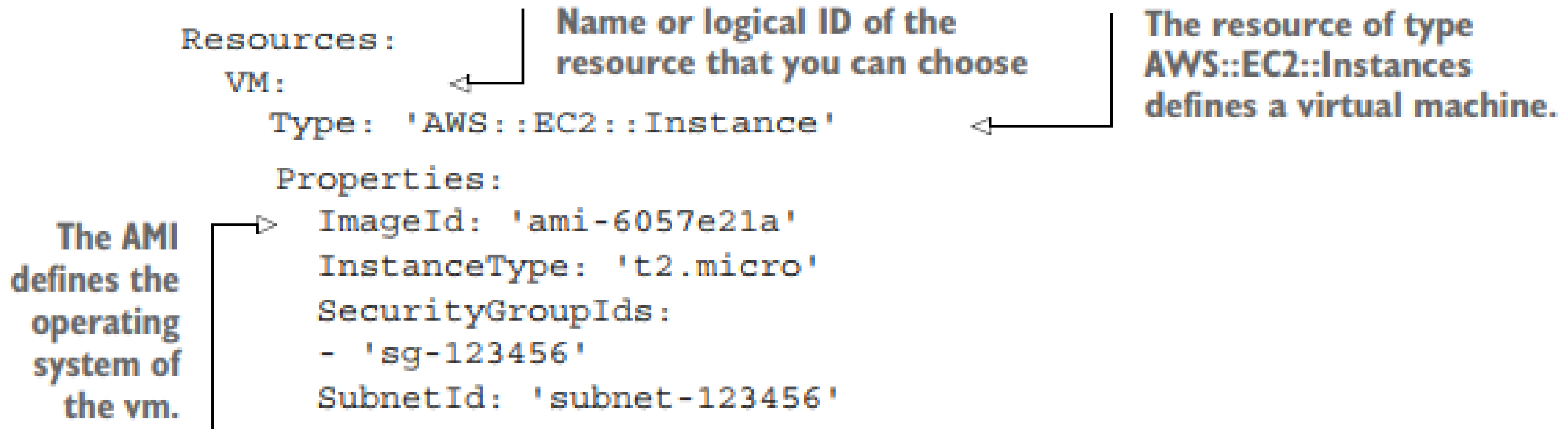


```
Resources:
  VM:
    Type: 'AWS::EC2::Instance'
    Properties:
      # [...]
```

Name or logical ID of the resource that you can choose
The resource of type AWS::EC2::Instances defines a virtual machine.
Properties needed for the type of resource

CloudFormation Template

- When defining resources, you need to know about the type and that type's properties. In this book, you'll get to know a lot of resource types and their respective properties. An example of a single EC2 instance appears in the following code snippet

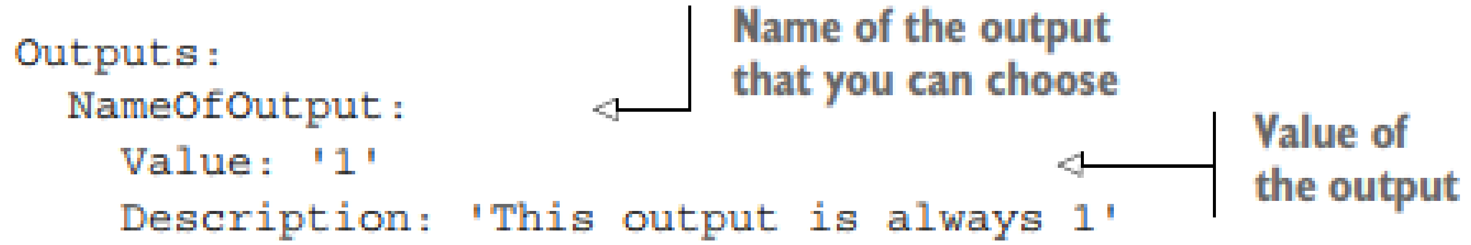


```
Resources:
  VM:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: 'ami-6057e21a'
      InstanceType: 't2.micro'
      SecurityGroupIds:
      - 'sg-123456'
      SubnetId: 'subnet-123456'
```

Name or logical ID of the resource that you can choose
The resource of type AWS::EC2::Instances defines a virtual machine.
The AMI defines the operating system of the vm.

CloudFormation Template

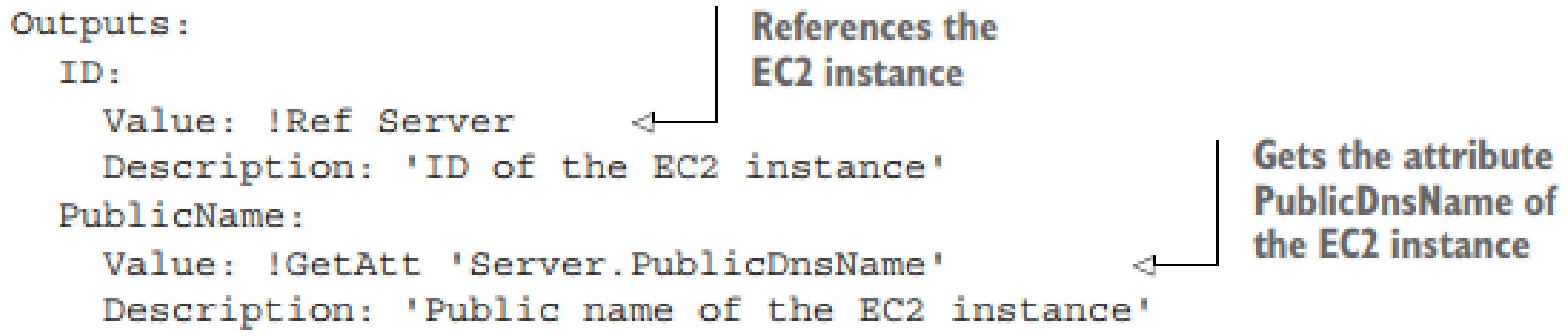
- A CloudFormation template’s **output** includes at least a name (like parameters and resources) and a value, but we encourage you to add a description as well, as illustrated in the next listing. You can use outputs to pass data from within your template to the outside.



```
Outputs:
  NameOfOutput:
    Value: '1'
    Description: 'This output is always 1'
```

Name of the output that you can choose

Value of the output



```
Outputs:
  ID:
    Value: !Ref Server
    Description: 'ID of the EC2 instance'
  PublicName:
    Value: !GetAtt 'Server.PublicDnsName'
    Description: 'Public name of the EC2 instance'
```

References the EC2 instance

Gets the attribute PublicDnsName of the EC2 instance