AWS - Why bother?

A Developers perspective

By Michael Fynes

What I am going to talk about

- Intro to New Things Co
- Why bother?
- Where to begin?
- Key terms
- Beginning to begin
- Cloud native approach
- Infrastructure as code
- User/role management
- Further learning
- Q & A

You build it, you run it
- Werner Vogels

Why bother?

“In the beginning there was darkness…”

- DevOps is a growing field but what is it?
- Cannot always be a dedicated person or team
- This talk is for developers looking to start

Key terms

- EB - Elastic Beanstalk (not to be confused with EBS)
- EC2 - Elastic Compute Generation 2
- RDS - Relational Database Services
- VPC - Virtual Private Cloud
- IAM - Identity & Access Management
- Load Balancer - Balances the load

Where do I begin?

- Elastic Beanstalk collects relevant web service resources
- Makes setting up small environments simple
- Plenty of learning materials
- Lightsail might be an alternative

Beginning to begin

1. Create your Elastic Beanstalk App by creating first environment
2. Create security keys and IAM profile for app
3. Change capacity to load balanced
4. Switch to Application load balancer
5. Setup second service and connect the two
6. Profit

What does any of that actually mean?

Creating your first Beanstalk application

- Pick a name
- Create environment / application (what kind of applications are supported, examples)
- Configure all the things

Platform Java 8 running on 64bit Amazon Linux/2.7.5 Change platform configuration

Software

AWS X-Ray: disabled
Rotate logs: disabled (default)
Log streaming: disabled (default)
Environment properties: 4
GRADLE_HOME, JAVA_HOME, M2, M2_HOME

Modify

Instances

EC2 instance type: t2.micro
EC2 image ID: ami-03e99fd807d2e7132
Root volume type: container default
Root volume size (GB): container default
Root volume IOPS: container default
Security groups: *none*

Modify

Capacity

Environment type: single instance

Modify

Load balancer

This configuration does not contain a load balancer.

Rolling updates and deployments

Deployment policy: All at once
Rolling updates: disabled

Modify

Security

Service role: *autogenerated*
Virtual machine key pair: --
Virtual machine instance profile: *autogenerated*

Modify

Monitoring

Health reporting system: Enhanced
Ignore HTTP 4xx: disabled
Health event log streaming: disabled

Modify

Notifications

Email address: --

Modify

Network

This environment is not part of a VPC.

Modify

Database

Engine: --
Instance class: --
Storage (GB): --
Multi-AZ: --

Modify

Tags

Tags: *none*

Modify

Configuring your environment

Software

- Logging options
- Environmental variables
- AWS X-Ray

Capacity

- Availability zones
- Scaling type
- Scaling triggers

Load balancer

- Ports to listen to
- Rules
- Health check pattern

Security

- Service role
- EC2 Key Pair
- IAM profile

Software

AWS X-Ray: disabled
Rotate logs: disabled (default)
Log streaming: disabled (default)
Environment properties: 5
GRADLE_HOME, JAVA_HOME, M2, M2_HOME, SERVER_PORT

Modify

Instances

EC2 instance type: t2.micro
EC2 image ID: ami-03e99fd807d2e7132
Monitoring interval: 5 minute
Root volume type: container default
Root volume size (GB): container default
Root volume IOPS: container default
Security groups: sg-0a752f770292f0230

Modify

Capacity

Environment type: load balancing, auto scaling
Availability Zones: Any
Instances: 1–4

Modify

Load balancer

Load balancer type: application
Listeners: 2
Processes: 1
Rules: 2

Modify

Rolling updates and deployments

Deployment policy: All at once
Rolling updates: disabled
Health check: enabled

Modify

Security

Service role: aws-elasticbeanstalk-service-role
Virtual machine key pair: aws-meetup-demo
Virtual machine instance profile: aws-elasticbeanstalk-ec2-role

Modify

Monitoring

Health reporting system: Enhanced
Ignore HTTP 4xx: disabled
Health event log streaming: disabled

Modify

Managed updates

Managed updates: disabled

Modify

Notifications

Email address: --

Modify

Network

This environment is not part of a VPC.

Database

Engine: --
Instance class: --
Storage (GB): --
Multi-AZ: --

Modify

What is this setup for

- Great for experimentation
- Not great for production (clicking leads to mistakes)
- Default VPC is not suitable for RDS data

The Cloud Native Approach

- Move quickly
- Small, easily reversible steps
- Stateless applications
- Disposable
- Environmentally independent
- Robust

Infrastructure as code

- Easy to reproduce in other environments and accounts
- Makes splitting production from test or development environments easy
- Usually allow a preview of what changes do
- Terraform uses HCL, can define entire infrastructure, provider agnostic
- Cloud Formation uses JSON or YAML, can define entire infrastructure, AWS only
- Ansible, can define servers and provision them, cannot define things like networks

Terraform example

Snippet HCL

```
resource "aws_s3_bucket" "newthings" {
  bucket = "newthings"
  acl    = "private"
  versioning {
      enabled = true
  }
  tags {
    Name        = "newthings"
    Environment = "dev"
  }
}
```

CloudFormation example

Snippet YAML

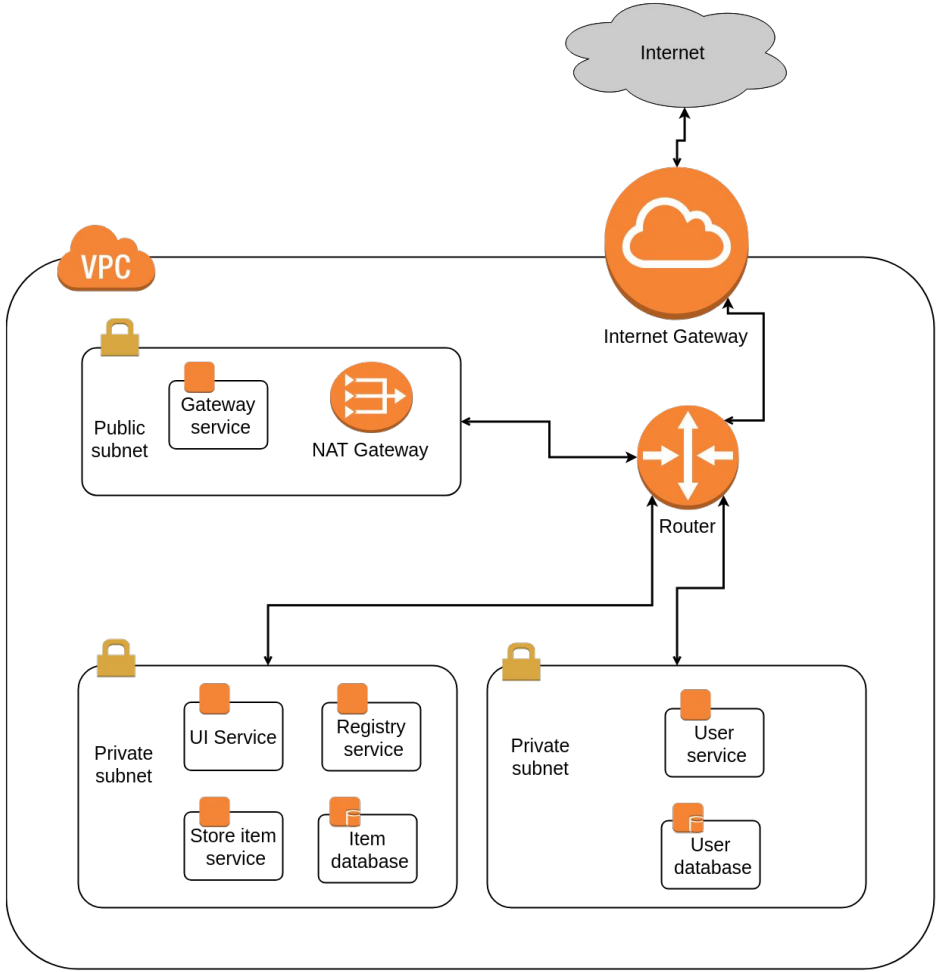
```
Resources:
     NewThingsBucket:
         Type: AWS::S3::Bucket
         Properties:
           BucketName: newthings
           AccessControl: private
           VersioningConfiguration:
             Status: Enabled
           Tags:
           - Key: Name
             Value: newthings
           - Key: Environment
             Value: dev
```

VPCs and their setup

- Virtual Private Cloud
- Software defined
- Helps isolate services
- Multi availability zones for High Availability

Simplified VPC diagram

- Gateway used to access private services
- Private customer information separated
- Flexibility like this useful due to GDPR
- See docs for more detailed examples



Final thoughts

- Can no longer just be “somebody else's problem”
- Hugely increases your value to your company and to others
- This is just first step down the rabbit hole
- Don't be afraid to experiment!

Further learning : General cloud tools

- VPCs
- Load balancers
- Docker
- Kubernetes
- Terraform, Ansible
- Serverless
- ELK stack- ElasticSearch, Logstash, Kibana

Further Learning : AWS Specific Tools

- CloudFront, Route 53 and other domain and routing tools
- EKS - Managed Kubernetes in AWS
- S3 - File storage in AWS
- DynamoDB - NoSQL DB
- SES - Simple Email Service
- Lambda - Serverless functions
- Fargate - Serverless containers
- CloudFormation - AWS Infra as code tool

Q & A


ITS A WRAP!

Links and guides used

- AWS Acronyms : https://geekflare.com/aws-related-acronyms/
- NodeJS tutorial breaking monolith to microservices : https://aws.amazon.com/getting-started/projects/break-monolith-app-microservices-ecs-docker-ec2/
- AWS VPC Basics : https://www.youtube.com/watch?v=bGDMeD6kOz0
- AWS VPC docs : https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Networking.html
- Explanation of Load Balancers : https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html
- Infra as code tools : https://www.thorntech.com/2018/04/15-infrastructure-as-code-tools/
- CloudFormation snippets : https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/CHAP_TemplateQuickRef.html
- Terraform modules : https://registry.terraform.io/
- Serverless learning:
    - https://serverless.com/blog - Serverless Framework
    - https://github.com/awslabs/serverless-application-model - AWS SAM
    - https://d1.awsstatic.com/whitepapers/architecture/AWS-Serverless-Applications-Lens.pdf