Scalable WordPress in AWS – Elastic Beanstalk

Stephen J. Butler, Technology Services
sbutler1@Illinois

Spring 2017
Tools

• Terraform
  • https://www.terraform.io/

• UIUC AWS
  • https://aws.illinois.edu/
  • aws-support@illinois.edu

• Labs
  • https://github.com/sbutler/itpf-sp2017-wordpress-terraform
  • https://github.com/sbutler/itpf-sp2017-wordpress-wordpress
Shared Data - RDS

- AWS Relational Database Service (RDS)
  - MySQL, MariaDB, or Aurora
  - PostgreSQL or Aurora
  - Oracle, MS SQL Server
- Managed updates, backups
- Hot standby and read replicas

<table>
<thead>
<tr>
<th>MariaDB Ohio</th>
<th>Single</th>
<th>Multi-AZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>db.t2.micro</td>
<td>$0.017/hr ($12.24)</td>
<td>$0.034/hr ($24.48)</td>
</tr>
<tr>
<td>db.t2.medium</td>
<td>$0.068/hr ($48.96)</td>
<td>$0.136/hr ($97.92)</td>
</tr>
<tr>
<td>db.m4.large</td>
<td>$0.175/hr ($126.00)</td>
<td>$0.350/hr ($252.00)</td>
</tr>
<tr>
<td>GP Storage</td>
<td>$0.115 per GB-mo</td>
<td>$0.230 per GB-mo</td>
</tr>
</tbody>
</table>
Shared Data - RDS

- MySQL vs MariaDB vs Aurora
  - MySQL: official release, most stable
  - MariaDB: community release, newer features and bug fixes
  - Aurora: AWS optimized (x5 performance) with MySQL compatibility
- Optional encryption of storage
- Use InnoDB tables!
Shared Data - RDS

- Backups
  - Snapshots
  - Momentarily freezes I/O
- Maintenance
  - Major or minor updates
  - Configuration changes
  - Anything that requires a restart
Shared Data - RDS

- Example includes a MariaDB instance
  - rds-wordpress.tf
- Refactor to share an instance
Shared Data - EFS

- AWS Elastic File System (EFS)
  - NFSv4.1 protocol
  - Unlimited storage
- Highly available within a region, across AZ's
- Not automatically backed up
- Pricing (Ohio): $0.30 per GB-mo
Shared Data - EFS

- Baseline rate based on amount of stored data
  - 50 MiB/s per TiB; 50 KiB/s per GiB
  - 1 GiB -> 50 KiB/s
  - 500 GiB -> 25 MiB/s
  - 10 TiB -> 500 MiB/s

- Burst Rates
  - < 1TiB -> 100 MiB/s
  - Additional 100 MiB/s burst per TiB stored

- Initial burst credit of 2.1 TiB
Shared Data - EFS

- Example includes a single EFS
  - efs-wordpress.tf
- With a clever directory structure, possible to share an instance
Web Hosting - EB

• AWS Elastic Beanstalk (EB)
  • Tomcat, .NET on IIS, PHP, Python, Ruby, Docker, etc
  • Abstracts the infrastructure for web applications
• Load balancing, multi-az, scaling, health monitoring, maintenance
• Application deployment and updating

<table>
<thead>
<tr>
<th>Ohio</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2.micro</td>
<td>$0.012/hr ($8.64)</td>
</tr>
<tr>
<td>t2.medium</td>
<td>$0.047/hr ($33.84)</td>
</tr>
<tr>
<td>m4.large</td>
<td>$0.100/hr ($72.00)</td>
</tr>
<tr>
<td>ELB</td>
<td>$0.025/hr ($18.00)</td>
</tr>
</tbody>
</table>
Web Hosting - EB

• Application
  • collection of all the EB components

• Application Version
  • deployable package (code)

• Environment
  • infrastructure resources

• Environment Configuration
  • Settings/options that customize the environment

• Configuration Template
  • Settings/options that can be applied to multiple environments
Web Hosting – EB Env
Web Hosting – EB Env

• Managed Actions: updates performed by AWS
• Performs rolling updates (optional)
• !! EC2 instances created from the app version !!
  • Do not manually change an instance
  • Do not store app data on the instance (see: RDS, EFS, S3)
Web Hosting – EB Config

- Determines settings and options for environment components
  - VPC subnets, etc
  - EC2 instance types, security groups, etc
  - ELB configuration, health check, protocols, etc
  - ASG size, thresholds, policies, etc
  - Managed actions window

- Merged from multiple sources
  1. Direct environment settings
  2. Configuration templates
  3. Application Version .ebextensions
  4. Default values
Web Hosting - EB

- Example includes a single app, config template, and env
  - eb-wordpress.tf
- Customize DNS with Route53
- Potentially create more environments for dev, test, etc
WordPress - Lab

- Upload Application Version
  - Can use "make eb-appver" to generate a new one
- Deploy Application Version to environment
- Wait for update to complete
WordPress

• Create an Application Version (.zip) for WordPress
  • App files included in the .zip file
    • Everything in your standard WP install
    • wp-content/plugins; wp-content/themes
    • wp-content/mu-plugins*
  • User content stored in RDS or on EFS
    • wp-content/uploads; wp-content/cache; wp-content/w3tc-config
    • wp-content/blogs.dir*
• wp-config.php loads settings from EB
• Health Check
• Connect EFS
WordPress – Standard

- Start with `.zip` from https://wordpress.org
  - Remove `wp-content/uploads` directory
- Add plugins
- Add themes
- Extra: `.ebextensions/wpcli.config`
  - Installs `wp-cli` command
• Start with the wp-config-sample.php
• Remember: we added our WP settings as environment variables!
  • Use $_SERVER['XXX'] for DB and salt settings
• Customize whatever else you'd like
WordPress – Health Check

• **aws-eb-ping.php**
  • Simple PHP check
  • Does not check the health of WordPress itself

• **.ebextensions/healthchecks.config**
  • Modify the configuration of our EB Env
  • Configures the ELB and EB to check /aws-eb-ping.php
WordPress - EFS

- EB does not automatically connect to EFS
- Remember: how did we do it in the terraform lesson?
- .ebextensions/efs-mount.config
  - Writes EFS config files using EB environment
  - Container setup:
    - Mounts EFS
    - Symlinks user data directories
  - Also periodically updates .htaccess
Monitoring

- Instance logs in CloudWatch Logs
- SNS Topic to generate emails
  - EB publishes notifications
  - RDS publishes notifications
  - CloudWatch Alarms for other interesting metrics
Monitoring - EB and RDS

- EB: configure an SNS Topic
- RDS: configure event subscription
Monitoring - Alarms

- CloudWatch Alarms to monitor other important metrics
  - EFS I/O limit
  - EFS burst credit balance
  - RDS connections
Questions

• Stephen J. Butler
  • sbutler1@illinois.edu

• UIUC AWS
  • https://aws.illinois.edu/
  • aws-support@illinois.edu

• Terraform
  • https://www.terraform.io/

• Labs
  • https://github.com/sbutler/itpf-sp2017-wordpress-terraform
  • https://github.com/sbutler/itpf-sp2017-wordpress-wordpress
Scalable WordPress in AWS - Terraform

Stephen J. Butler, Technology Services
sbutler1@Illinois
Tools

• Terraform
  • https://www.terraform.io/

• UIUC AWS
  • https://aws.illinois.edu/
  • aws-support@illinois.edu

• Labs
  • https://github.com/sbutler/itpf-sp2017-terraform
Introduction – What is IaC?

Wikipedia - Infrastructure as Code

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Terraform - Infrastructure as Code

Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.
Introduction – Why IaC?

- Repeatable Deployments (dev, test, prod)
- Shareable deployment strategies
- Versioned configuration changes
- Predictable configuration changes
- Reduce infrastructure related errors and risk
Introduction – Terraform

• Open Source project for IaC
• Multiple platforms
• Multiple providers
• More flexible than CloudFormation
• (Faster than CloudFormation)
Terraform Layout

- Directory of ".tf" files
- Personal preference

<table>
<thead>
<tr>
<th>File/Subdirectory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>providers.tf</td>
<td>Setup for terraform providers</td>
</tr>
<tr>
<td>variables.tf</td>
<td>Variables used by the configuration</td>
</tr>
<tr>
<td>outputs.tf</td>
<td>Outputs for the configuration</td>
</tr>
<tr>
<td>files/</td>
<td>Static files referenced in a config</td>
</tr>
<tr>
<td>modules/</td>
<td>Local terraform modules*</td>
</tr>
<tr>
<td>scripts/</td>
<td>Local or remote scripts to run for provisioning*</td>
</tr>
<tr>
<td>templates/</td>
<td>Dynamic template files for the &quot;template&quot; provider</td>
</tr>
<tr>
<td>variables/</td>
<td>Variable (&quot;.tfvar&quot;) files to change a config</td>
</tr>
</tbody>
</table>
Providers

- Providers the backend for terraform components
  - aws, google, azurerm
  - mysql, http, archive, template_file
  - https://www.terraform.io/docs/providers/
- Examine "providers.tf"
Resources - Overview

- Something that exists in the infrastructure
  - EC2 instance; ELB; EB Environment; IAM Role; ...
  - [https://www.terraform.io/docs/providers/aws](https://www.terraform.io/docs/providers/aws)
- What terraform manages (Create, Update, Destroy)
- Must have a **type** and a unique **name**

```terraform
resource "aws_instance" "example" {
    # attributes and blocks
}
```
Resources - Create

- Example `instances.tf`
- Add attributes, plan, and apply

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami</td>
<td>ami-7bfcd81e</td>
</tr>
<tr>
<td>instance_type</td>
<td>t2.micro</td>
</tr>
<tr>
<td>key_name</td>
<td>workshop</td>
</tr>
</tbody>
</table>
Resources - Update

• Example **instances.tf**

• Add **tags** block for **Name** and **NetID**, plan, and apply.

```hcl
resource "aws_instance" "example" {
    # attributes
    tags {
        Name = "…"
        NetID = "…"
    }
}
```
Variables - Overview

• Why: greater reuse and flexibility
  • string: "abc", "1", "2"
  • list: [ "foo", "bar", "baz" ]
  • map*: { "foo" = "a" }

• Must be declared, and have a unique name

```javascript
variable "my_example" {
  type = "string"
  description = "Variable named my_example"
}

# Referenced "$\{\text{var.NAME}\}"
some_attribute = "$\{\text{var.my_example}\}"
```
Variables - Sources

- **Prompt**
- **Command Line:**
  - `terraform plan -var foo=1 -var bar=2`
- **Files**
  - `terraform plan -var-file variables/project.tfvars`
- **Can specify multiples; uses order they were given**
Variables - Lab

- **variables.tf**: add string variables `netid`, `key_name`, `instance_type`
- **instances.tf**: modify to use appropriate variables

```hcl
variable "netid" {
  type = "string"
  description = "Contact for resources in this config"
}

resource "aws_instance" "example" {
  # attributes
  tags {
    Name = "…"
    NetID = "$\{\text{var.netid}\}"
  }
}
```
Variables – terraform.tfvars

- Will automatically load terraform.tfvars
  - Implied in ever command: -var-file terraform.tfvars
- Uncomment **instance_type** and **key_name**
- Add **netid**
Outputs - Overview

- Easily display useful results
- Must be declared and have a unique **name**
- Value is an attribute on resources, data, variables, etc

```yaml
output "my_resource_output" {
  value = "${TYPE.NAME.ATTRIBUTE}" 
}

# Example: "${aws_instance.example.id}"
Outputs - Lab

- Example: `outputs.tf`
  - `aws_instance.example.public_ip`
  - `aws_instance.example.id`

```hcl
output "example_id" {
  value = "${aws_instance.example.id}" 
}
```
Data - Overview

- Why: gathers information for use in configs
  - Remote sources: ACM certs; AMI; EIP; etc
  - Local sources: rendered templates, IAM Policy Docs, etc
- Referenced by other data blocks or resources
- Validate values specified in variables
- Data blocks have a **type** and a unique **name**

```hcl
data "aws_subnet" "public" {
  id = "${var.public_subnet}"
}

# Referenced "${data.TYPE.NAME.ATTRIBUTE}"
some_attribute = "${data.aws_subnet.public.id}"```

Spring 2017
Data - Lab

• **Example: instances.tf**
  - Add data block to find the latest Amazon Linux AMI id
  - Run terraform plan to see what happens

```terraform
data "aws_ami" "amazon_linux" {
  most_recent = true
  owners = [ "amazon" ]
  filter {name = "name"; values = ["amzn-ami-*-gp2"]}
  filter {name = "virtualization-type"; values = ["hvm"]}
  filter {name = "architecture"; values = ["x86_64"]}
}

resource "aws_instance" "example" {
  ami = "${data.aws_ami.amazon_linux.id}"  # attributes and blocks
}
```
Resources - Update

• Stop! Changing the AMI will recreate the resource
• Terraform shows this with "-/+" instead of "~".
• Preventing recreation on AMI changes:

```hcl
data "aws_ami" "amazon_linux" {
    most_recent = true
    # ...
}

resource "aws_instance" "example" {
    ami = "${data.aws_ami.amazon_linux.id}"
    # attributes and blocks
    lifecycle { ignore_changes = [ "ami" ] }
}
```
Templates - Overview

• Generate strings to be used for resources, other data, etc
• Templates are a kind of data; accepts vars to use in the template
• Best: store template string in its own file
data "template_file" "hello_world" { 
    template = "${file("templates/hello_world.txt")}" 
    vars { 
        name = "John Doe" 
        message = "How are you doing today?"
    } 
}

# Referenced "${data.template_file.NAME.rendered}" 
an_attr = "${data.template_file.hello_world.rendered}" 

Hello, ${name}! 

${message}
Hello, John Doe!

How are you doing today?
Cloud-Init - Overview

• Cloud-Init is a way to customize your instances on first boot
  • Boot hooks
  • Configuration: timezone, packages, files, run commands
  • First boot scripts
  • Upstart scripts
  • https://cloudinit.readthedocs.io/en/latest/

• MIME Document specified in the instance UserData
  • Can be built using a terraform data block
  • Particularly useful with templates!
Cloud-Init - Lab

• Use cloud-init to connect your instance to EFS; using Tech Services developed script

• Write an EFS config file
  • /etc/opt/uiuc-techservices/efs/sharedfs.conf
  • Contains settings for mounting the EFS volume
    • efs_filesystem_id
    • mount_target (optional - /mnt/sharedfs)
    • nfs_options (optional – AWS defaults)

• Run a first boot script (files/efs.sh)
  • Reads all config files, mounts them, and updates fstab
• Configuration template

```yaml
data "template_file" "cloudinit_config" {
    template = "${file("templates/cloudinit_config.yml")}" 
    vars {
        efs_id = "${var.sharedfs_id}" 
    } 
}
```

```yaml
packages:
    - nfs-utils

write_files:
    - path: /etc/opt/uiuc-techservices/efs/sharedfs.conf
      owner: root:root
      permission: '0644'
      content: efs_filesystem_id=${efs_id}
```
Cloud-Init – Lab

```hcl
data "template_cloudinit_config" "userdata" {
    part {
        content_type = "text/cloud-config"
        content = "${data.template_file.cloudinit_conf.rendered}"
    }
    part {
        content_type = "text/x-shellscript"
        filename = "efs.sh"
        content = "${file("files/efs.sh")}"
    }
}

resource "aws_instance" "example" {
    # attributes
    user_data = "${data.template_cloudinit_config.userdata.rendered}"
}
```
Cloud-Init - Lab

- Run plan and check the output
  - Fix errors
  - Will require recreating the instance
- Run apply
- SSH and check that /mnt/sharedfs exists and is writable
Questions

• Stephen J. Butler
  • sbutler1@illinois.edu

• UIUC AWS
  • https://aws.illinois.edu/
  • aws-support@illinois.edu

• Terraform
  • https://www.terraform.io/

• Labs
  • https://github.com/sbutler/itpf-sp2017-terraform