**Five Talent Software – Operations Runbook**

**Contents**

Table of Contents

[Section 1 Important Information Start Here 6](#_Toc534812334)

[Section 1.1 Five Talent Contact Info 6](#_Toc534812335)

[Section 1.2 Alert Acknowledgement Process 6](#_Toc534812336)

[Section 1.3 Connecting to AWS Environment 6](#_Toc534812337)

[Section 1.4 Notes 6](#_Toc534812338)

[Section 2 Alert Matrices 6](#_Toc534812339)

[Section 2.1 Windows Server Alerts 6](#_Toc534812340)

[Section 2.2 Linux alerts 6](#_Toc534812341)

[Section 2.3 Network Alerts 6](#_Toc534812342)

[Section 2.4 Storage Alerts 6](#_Toc534812343)

[Section 2.5 Database Alerts 6](#_Toc534812344)

[Section 2.6 Service Alerts 6](#_Toc534812345)

[Section 2.7 Application alerts 6](#_Toc534812346)

[Section 2.8 Virtual Infrastructure 6](#_Toc534812347)

[Section 3 Troubleshooting / Resolution Steps 6](#_Toc534812348)

[Section 3.1 Windows Server Alerts 6](#_Toc534812349)

[Section 3.2 Linux alerts 6](#_Toc534812350)

[Section 3.3 Network Alerts 6](#_Toc534812351)

[Section 3.4 Storage Alerts 6](#_Toc534812352)

[Section 3.5 Database Alerts 6](#_Toc534812353)

[Section 3.6 Service Alerts 6](#_Toc534812354)

[Section 3.7 Application alerts 6](#_Toc534812355)

[Section 3.8 Virtual Infrastructure 6](#_Toc534812356)

[Section 4 Escalation Processes 6](#_Toc534812357)

[Section 5 Network Diagrams 6](#_Toc534812358)

[Section 6 Monitoring System 6](#_Toc534812359)

[Section 6.1 CloudWatch Alarms (From the FTS Wiki) 6](#_Toc534812360)

[Section 6.2 Monit Software 8](#_Toc534812361)

[Section 6.2.1 Installing Monit Software on Linux (Debian) 8](#_Toc534812362)

[Section 7 Resource List 8](#_Toc534812363)

[Section 7.1 Networks 8](#_Toc534812364)

[Section 7.2 Storage 8](#_Toc534812365)

[Section 7.3 Database Instances / RDS 8](#_Toc534812366)

[Section 7.4 Lambda Functions 8](#_Toc534812367)

[Section 7.5 EC2 Servers 8](#_Toc534812368)

[Section 8 Environment Overview 8](#_Toc534812369)

[Section 8.1 Tagging Categories 8](#_Toc534812370)

[Section 8.1.1 Technical Tags 9](#_Toc534812371)

[Section 8.1.2 Tags for Automation 9](#_Toc534812372)

[Section 8.1.3 Business Tags 9](#_Toc534812373)

[Section 8.2 Security Tags 9](#_Toc534812374)

[Section 9 Supporting SOPs 9](#_Toc534812375)

[Section 9.1 Change Management 9](#_Toc534812376)

[Section 9.1.1 CloudFormation 9](#_Toc534812377)

[Section 9.1.2 CloudFormer (Reverse engineer; create CloudFormationTemplates 9](#_Toc534812378)

[Section 9.1.3 Deploy and Use CloudFormer 12](#_Toc534812379)

[Section 9.2 Release Management 13](#_Toc534812380)

[Section 9.2.1 Using Git 13](#_Toc534812381)

[Section 9.2.2 Create a new Git repository in Code Commit 13](#_Toc534812382)

[Section 9.2.3 Create a new local repository and connect to a Code Commit repository 15](#_Toc534812383)

[Section 9.3 Patching and Updates 18](#_Toc534812384)

[Section 9.3.1 MOP Upgrade an EC2 instance type 18](#_Toc534812385)

[Section 9.3.2 MOP Review and lock down NACLS and security groups to only required Ports 21](#_Toc534812386)

[Section 9.4 Amazon Web Services 23](#_Toc534812387)

[Section 9.4.1 Cloud Trail setup for Multiple A 23](#_Toc534812388)

[Section 9.4.2 CloudTrail Logging 23](#_Toc534812389)

[Section 9.5 Citrix 23](#_Toc534812390)

[Section 9.6 VMware 23](#_Toc534812391)

[Section 9.7 Linode (Another Cloud Provider) 23](#_Toc534812392)

[Section 9.7.1 Clone a Linode Instance 24](#_Toc534812393)

[Section 9.7.2 MOP Reproduce a Linode Instance for CrimeDex 24](#_Toc534812394)

[Section 9.8 Windows 25](#_Toc534812395)

[Section 9.9 Storage 25](#_Toc534812396)

[Section 9.10 Linux 25](#_Toc534812397)

[Section 9.11 Network Devices 25](#_Toc534812398)

[Section 9.12 Application 25](#_Toc534812399)

[Section 9.12.1 Service or system overview 25](#_Toc534812400)

[Section 9.12.2 Business overview 25](#_Toc534812401)

[Section 9.12.3 Technical overview 26](#_Toc534812402)

[Section 9.12.4 Service Level Agreements (SLAs) 26](#_Toc534812403)

[Section 9.12.5 Service owner 26](#_Toc534812404)

[Section 9.12.6 Contributing applications, daemons, services, middleware 26](#_Toc534812405)

[Section 9.12.7 System characteristics 26](#_Toc534812406)

[Section 9.12.8 Hours of operation 26](#_Toc534812407)

[Section 9.12.9 Data and processing flows 26](#_Toc534812408)

[Section 9.12.10 Infrastructure and network design 26](#_Toc534812409)

[Section 9.12.11 Resilience, Fault Tolerance (FT) and High Availability (HA) 27](#_Toc534812410)

[Section 9.12.12 Throttling and partial shutdown 27](#_Toc534812411)

[Section 9.12.13 Expected traffic and load 27](#_Toc534812412)

[Section 9.12.14 Environmental differences 27](#_Toc534812413)

[Section 9.12.15 Tools 27](#_Toc534812414)

[Section 9.12.16 Required resources 28](#_Toc534812415)

[Section 9.12.17 Required resources - compute 28](#_Toc534812416)

[Section 9.12.18 Required resources - storage 28](#_Toc534812417)

[Section 9.12.19 Required resources - database 28](#_Toc534812418)

[Section 9.12.20 Required resources - metrics 28](#_Toc534812419)

[Section 9.12.21 Required resources - logging 28](#_Toc534812420)

[Section 9.12.22 Required resources - other 28](#_Toc534812421)

[Section 9.12.23 Security and access control 28](#_Toc534812422)

[Section 9.12.24 Password and PII security 28](#_Toc534812423)

[Section 9.12.25 Ongoing security checks 28](#_Toc534812424)

[Section 9.12.26 System configuration 28](#_Toc534812425)

[Section 9.12.27 Configuration management 28](#_Toc534812426)

[Section 9.12.28 Secrets management 28](#_Toc534812427)

[Section 9.12.29 System backup and restore 29](#_Toc534812428)

[Section 9.12.30 Backup requirements 29](#_Toc534812429)

[Section 9.12.31 Backup procedures 29](#_Toc534812430)

[Section 9.12.32 Restore procedures 29](#_Toc534812431)

[Section 9.12.33 Monitoring and alerting 29](#_Toc534812432)

[Section 9.12.34 Log aggregation solution 29](#_Toc534812433)

[Section 9.12.35 Log message format 29](#_Toc534812434)

[Section 9.12.36 Events and error messages 29](#_Toc534812435)

[Section 9.12.37 Metrics 29](#_Toc534812436)

[Section 9.12.38 Health checks 30](#_Toc534812437)

[Section 9.12.39 Operational tasks 30](#_Toc534812438)

[Section 9.12.40 Deployment 30](#_Toc534812439)

[Section 9.12.41 Batch processing 30](#_Toc534812440)

[Section 9.12.42 Power procedures 30](#_Toc534812441)

[Section 9.12.43 Routine and sanity checks 30](#_Toc534812442)

[Section 9.12.44 Troubleshooting 30](#_Toc534812443)

[Section 9.12.45 Maintenance tasks 30](#_Toc534812444)

[Section 9.12.46 Patching 30](#_Toc534812445)

[Section 9.12.47 Daylight-saving time changes 31](#_Toc534812446)

[Section 9.12.48 Data cleardown 31](#_Toc534812447)

[Section 9.13 Log rotation 31](#_Toc534812448)

[Section 9.13.1 Failover and Recovery procedures 31](#_Toc534812449)

[Section 9.13.2 Failover 31](#_Toc534812450)

[Section 9.13.3 Recovery 31](#_Toc534812451)

[Section 9.13.4 Troubleshooting Failover and Recovery 31](#_Toc534812452)

[Section 9.14 Liquid Planner 31](#_Toc534812453)

[Section 9.15 Okta 32](#_Toc534812454)

[Section 9.16 Amazon Workdocs 32](#_Toc534812455)

[Section 10 Backup / Restore 32](#_Toc534812456)

[Section 10.1 MOP – Create EBS Backup strategy using EBS Lifecycle Policies 32](#_Toc534812457)

[Section 11 Antivirus and Related Security 33](#_Toc534812458)

[Section 12 Systems Provisioning 33](#_Toc534812459)

[Section 12.1 Create Workstation for TriComm 33](#_Toc534812460)

[Section 12.2 Create a new single instance Debian WordPress Server 34](#_Toc534812461)

[Section 13 User Provesioning to AWS 36](#_Toc534812462)

[Section 14 Key Contact Information 36](#_Toc534812463)

[Section 15 AWS IAM Users, Groups, Roles 36](#_Toc534812464)

[Section 16 Appendix 36](#_Toc534812465)

[Section 16.1 Customer Processes to be known 36](#_Toc534812466)

[Section 16.2 Other environment-specific information for reference 36](#_Toc534812467)

[Section 16.2.1 AWS Partner Portal 36](#_Toc534812468)

[Section 16.3 Index of Terms 36](#_Toc534812469)

[Section 16.4 Section Templates 36](#_Toc534812470)

[Section 16.4.1 Server Template 36](#_Toc534812471)

[Section 16.4.2 Device Grid 36](#_Toc534812472)

[Section 16.4.3 Procedure Template 37](#_Toc534812473)

[Section 16.4.4 MOP Template 38](#_Toc534812474)

[Section 17 Document Revision History 39](#_Toc534812475)

[Section 18 Headings Check 39](#_Toc534812476)

[Section 18.1.1 Heading 3 39](#_Toc534812477)

[Section 19 Headings Check 39](#_Toc534812478)

# Important Information Start Here

## Five Talent Contact Info

## Alert Acknowledgement Process

## Connecting to AWS Environment

## Notes

# Alert Matrices

## Windows Server Alerts

## Linux alerts

## Network Alerts

## Storage Alerts

## Database Alerts

## Service Alerts

## Application alerts

## Virtual Infrastructure

# Troubleshooting / Resolution Steps

## Windows Server Alerts

## Linux alerts

## Network Alerts

## Storage Alerts

## Database Alerts

## Service Alerts

## Application alerts

## Virtual Infrastructure

# Escalation Processes

# Network Diagrams

# Monitoring System

## CloudWatch Alarms (From the FTS Wiki)

* [Alarm Naming](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#alarm_naming)
  + [Monitoring Scripts](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#monitoring_scripts)
    - [Linux](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#linux)
    - [Windows](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#windows)
  + [t2.micro](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#t2micro)
  + [t2.small](https://wiki.fivetalent.com/amazon_web_services:alarm_naming#t2small)

#### Alarm Naming

ClientName-ServiceName-MetricType-MetricValue/ConsecutivePeriods/Interval

Example: FiveTalent-WebServer-CPU-50/5/1 (CPU over 50 percent for 5 consecutive periods of 1 minute)

Edit

#### Monitoring Scripts

Edit

#### Linux

<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/mon-scripts-perl.html> (use the awscreds config file)

Cronjob Command:

~/aws-scripts-mon/mon-put-instance-data.pl --mem-util --mem-used --mem-avail --swap-util --swap-used --disk-path=/ --disk-space-util --disk-space-avail --from-cron

Edit

#### Windows

<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/mon-scripts-powershell.html>

Prerequisite Command:

Set-ExecutionPolicy Unrestricted

Cronjob Command:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "C:\CloudWatch\mon-put-metrics-mem.ps1 -mem\_util -mem\_used -mem\_avail -page\_util -page\_used -page\_avail -from\_scheduler -aws\_credential\_file C:\CloudWatch\awscreds.conf"

Cronjob Command:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "C:\CloudWatch\mon-put-metrics-disk.ps1 -disk\_space\_util -disk\_space\_avail -disk\_drive C: -from\_scheduler -aws\_credential\_file C:\CloudWatch\awscreds.conf"

Cronjob Command:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "C:\CloudWatch\mon-put-metrics-perfmon.ps1 -processor\_queue -pages\_input -from\_scheduler -aws\_credential\_file C:\CloudWatch\awscreds.conf"

Edit

#### t2.micro

Cronjob Interval: 20 minutes

* Unordered List ItemCPU >= 50/5/1
* CPUCreditBalance < 50/5/1
* StatusCheck (ANY) 5/1
* MemoryUtilization >= 100/5/1
* DiskSpace < 2(GB)/5/300

Edit

#### t2.small

Cronjob Interval: 20 minutes

* CPU >= 50/5/1
* CPUCreditBalance < 100/5/1
* StatusCheck (ANY) 5/1
* MemoryUtilization >= 100/5/1
* DiskSpace < 2(GB)/5/300

## Monit Software

Monit is a linux based monitor process that can check processes, files, programs, the system, and take proactive action. <https://mmonit.com/monit>

### Installing Monit Software on Linux (Debian)

# Resource List

## Networks

## Storage

## Database Instances / RDS

## Lambda Functions

## EC2 Servers

# Environment Overview

## Tagging Categories

REFERENCE: <https://aws.amazon.com/answers/account-management/aws-tagging-strategies/>

### Technical Tags

Name –

Applicaction ID

Application Role

Cluster –

Environment – dev, test, prod

Version

### Tags for Automation

Date/Time – used to identify the date or time a resource should be started, stopped, deleted, or rotated

Opt In/Opt out – used to indicate whether a resource should be automatically included in an automated activity such as starting, stopping, or resizing instances

Security – Used to determine requirements such as encryption or enabling of VPC Flow Logs, and also to identify route tables or security groups that deserve extra scrutiny

### Business Tags

Owner –

Cost Center/Business Unit

Customer

Project

## Security Tags

Confidentiality – An identifier for the specific data-confidentiality level a resources supports

Compliance – An identifier for workloads designed to adhere to specific compliance requirements

# Supporting SOPs

## Change Management

### CloudFormation

### CloudFormer (Reverse engineer; create CloudFormationTemplates

#### Setup CloudFormer

##### Summary

This will prepare AWS CloudFormation to utilize the CloudFormer template to reverse engineer an environment. NOTE: This process creates an EC2 Instance that will be billed.

##### Reference

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-cloudformer.html>

<https://aws.amazon.com/blogs/devops/building-aws-cloudformation-templates-using-cloudformer/>

##### Input Variables

An IAM Account with CloudFormation access to the target environment

##### Validation

The stack is successfully created. Note the URL output of the EC2 instance

##### Procedure

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Log into the AWS Console and navigate to CloudFormation |  |
|  | Click create stack button |  |
|  | Choose the sample template – CloudFormer.  Click Next |  |
|  | Enter a Stackname, username, and credential (FiveTalent!) The user name and cred are to the new EC2 instance that runs the CloudFormer process  NOTE: Use the Create New VPC option, not the default shown in screenshot  Click Next |  |
|  | Select Options as needed  Click Next |  |
|  | Review all settings |  |
|  | Click create – WARNING, This will create the running stack and a new EC2 instance. Good idea to notify the client before clicking create. |  |
|  | The Stack is built |  |
|  | Wait for Completion. If errors/rollback happens, troubleshoot and rerun |  |
|  | Look on the Output tab for the custom URL |  |
|  | Procedure Complete |  |

* + - 1. *Notes*

### Deploy and Use CloudFormer

#### Summary

#### Reference

#### Input Variables

The URL for the CloudFormer EC2 instance created above.

#### Validation

#### Procedure

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Access the CloudFormer URL from a browser. Log in with the credentials |  |
|  | Select the AWS region  Click Create Template |  |
|  | Enter a description, and optionally a tag name filter |  |
|  | Choose desired resource options on the next several pages for Networking, Compute, storage, etc. |  |
|  | Review all setting |  |
|  | Run the process, which creates the new JSON document |  |
|  | Save the JSON template (S3 bucket?) |  |
|  | Delete the stack. This will cleanup and terminate the EC2 instance |  |
|  |  |  |
|  | Procedure Complete. |  |

* + - 1. *Notes*

## Release Management

### Using Git

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ssh-windows.html>

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-https-unixes.html?icmpid=docs_acc_console_connect#setting-up-https-unixes-account>

#### Creating a local Git Repository

A New repo from an existing project

1. Go into the directory contain the project

2. Type git init

3. Type git add to add all of the relevant files

4. You'll probably want to create a .gigignore file right away, to indicate all of the files you don't want to track. Use git add .gitignore, too

5. Type git commit

Connect local to remote git repository in Code Commit

Git remote add origin ssh://APKAJQYIZRICVM4BTK5Q@git-codecommit.us-west-2.amazonaws.com/v1/repos/BrianKeithTEST

### Create a new Git repository in Code Commit

* + - 1. *Summary*

Code Commit is similar to github. A repository is created in AWS Code Commit, then used for source control and code check in/check out

* + - 1. *Reference*

<https://docs.aws.amazon.com/codecommit/latest/userguide/getting-started-cc.html?icmpid=docs_acc_console>

* + - 1. *Input Variables*

varRepositoryName = The name of the new repository, usually contains the client name, project name, and type/purpose

i.e. “RealTimeResearch\_RTRWeb\_CloudFormation”

* + - 1. *Validation*

The new repository is created and files can be added

* + - 1. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Log into the Five Talent AWS Sandbox account management console and drill down to the Code Commit dashboard |  |
|  | Click the **Create Repository** button |  |
|  | Add the repository name and description.  Click **Create Repository** button |  |
|  | Choose the appropriate options for CloudWatch notifications. |  |
|  | Click the **Create SNS Topic** button |  |
|  | Enter a new topic name and topic display name (10 characters)  Click the **Create** button |  |
|  | Click the **manage subscriptions** link |  |
|  | Add your email address. Look in your inbox for a confirmation email |  |
|  | Click close. An initial file can be added via the console |  |
|  | Procedure Complete. |  |

* + - 1. *Notes*

### Create a new local repository and connect to a Code Commit repository

* + - 1. *Summary*

The procedure will create a new local folder as a git repository, and then pull the files from Code Commit to allow working on local copies. This assume git for windows has been installed and a Code Commit repository already exists

* + - 1. *Reference*

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ssh-windows.html>

<https://docs.aws.amazon.com/codecommit/latest/userguide/getting-started.html>

* + - 1. *Input Variables*

varReposityEndpoint

* + - 1. *Validation*

File are retrieved and added to the local repository

* + - 1. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | On the local workstation (or AWS Workplace workstation) create a new local project folder. Recommend keeping it a short path name |  |
|  | Open the Git Bash client and navigate to created folder |  |
|  | Command$ **git init**  This will create the local repository |  |
|  | Set the origin to connect local to remote git repository in Code Commit  **Git clone**  **ssh://APKAJQYIZRICVM4BTK5Q@git-codecommit.us-west-2.amazonaws.com/v1/repos/Alexandria**  Note: See Jeff about getting a private key created  **Git clone -b <branch> <remote\_repo>**  **NOTE: be sure and include the SSH key name “APKAJ….@” portion on the clone request – see screen shot**  The command will ask for your private passphrase |  |
|  | Validate that files from the Code Commit repository now exist in the local directory |  |
|  | Git status  Git add .  Git commit -m ‘message’  Git Push |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

*Notes*

## Patching and Updates

### MOP Upgrade an EC2 instance type

**Client: RealTime Research. Upgrade Instance Type**

*\*\*\* Remediation: Upgrade to latest series (m5) and review family and size based on metrics*

*https://app.liquidplanner.com/space/71840/projects/show/46511838*

**Synopsis**: This procedure will upgrade an EC2 instance from an older type to a current type. This will generally save money as AWS charges more for older instance types, to encourage clients to upgrade.

**Environment**: AWS EC2

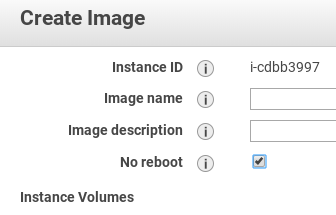
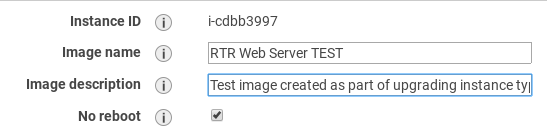
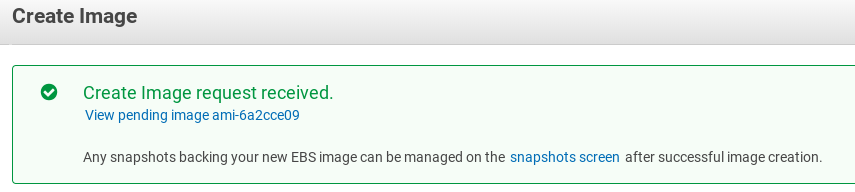
**Analysis**: From a WAFR review, older instance types are discovered and flagged for upgrade.

Notes and Compatibility for Resizing Instances:

EC2 Instance: N. California Region, Instance ID: I-cdbb399, Name: RTR Web Server. Type: m1.large, Target type: m5.large, key-name: ocs-rtr. Two Volumes: Root (vol-f4e440a0) and D on Web (vol-4d14ba19)

1. Virtualization type: Paravirtual (PV) or hardware virtual machine (HVM)
   1. Cannot load details for ami-a02015e5....
   2. The virtualization is HVM
2. Network: some types not supported in EC2-Classic
   1. The environment is EC2-VPC, (not classic)
3. Platform: 32 bit vs. 64 bit AMI (check EC2 console, show/hide Architecture column)
   1. The platform is x86\_64 (64 bit)
4. Enhanced Networking -
   1. Assumed not; need to check from command line
5. NVMe: Some types support NVMe block devices and need special drivers
   1. No – using a standard volume
6. Check for Elastic IPs to the instance.
   1. Yes – 54.215.14.20 (*will change if instance moved to Oregon Region)*
7. Check for EBS Root Volume vs. Instance Store Root Volume
   1. Two volumes, both EBS

**Resolution**:

1. Review the target instance for compatibility.
   1. completed
2. Check with client about taking the instance offline and/or stopping read/writes. "To create a snapshot for Amazon EBS volumes that serve as root devices, you should stop the instance before taking the snapshot."
   1. Maintenance Window ?
3. Create an Image of the EC2 instance. From the EC2/EBS console,
   1. Include Both Volumes (WARNING, be sure and check the box for NO REBOOT.
   2. 
   3. 
   4. 
   5. Process started at 2:17pm, finished at 3:31pm (*Will likely go faster if instance is shut down first. Need a large maintenance window.)*
4. Create a new Sandbox VPC using a cloudformation template
   1. CF\_RealTimeResearch\_Network\_V1.1.template.json
   2. Completed successfully
5. Create a new instance from the AMI with the new type. Create in sandboxVPC
   1. Select an m4.large instance type (m5.large not available, likely due to networking)
   2. Use the default vpc-f9e3d19d
   3. Use the default sg-ce5513a8
   4. Choose use GeneralPurpose SSD for boot volume
   5. NOTE: Security group, default, is open to the world
   6. NOTE: created new Key for testing; RTRtestkeyforFTS. Likely need to use ocs-rtr key for the production instance.
6. Create a new RDS instance in the SandboxVPC
   1. Using AWS console, navigate to RDS >> Snapshots
   2. Select the latest snapshot
   3. Click Actions >> Restore Snapshot
   4. For testing, used a db.t2.medium instance class
   5. DB Instance Identifier: db-100-SANDBOX
   6. VPC: RTRSandboxVPC
   7. Subnetgroup: Create new DB Subnet Group - FAIL, need an existing subnet group
   8. Public Accessibility – YES, may need to verify with client, is this needed?
7. Validation: Use a public IpV4 or DNS endpoint and allow client to test for functionality. Complex environments may require a separate sandbox VPC for testing.
   1. Successful connection to <https://54.183.183.7/index.html>
   2. WHAT RDS INSTANCE IS IN USE?? Need to verify first. Test Instance is shut down.
8. Send connection RDP file, and IP address to client for validation
9. After validation by client, schedule a cut over time to repeat the process. The existing instance should be shut down to prevent any further reads and writes.
10. Detach the Elastic IP
11. Create a new AMI
12. Create a new larger instance from the AMI
13. Attach the ElasticIP
14. Validation: Client can access and test the application
15. Delete any original testing AMIs/Instances
16. Take a final snapshot and delete the original instance. (Client Choice)

**Further Research**:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-resize.html>

### MOP Review and lock down NACLS and security groups to only required Ports

**Client**: **RealTime Research. Review NACLs and Security Groups**

*\*\*\* Remediation: Lock down NACLS and security groups to only required ports*

[*https://app.liquidplanner.com/space/71840/projects/show/46511758P*](https://app.liquidplanner.com/space/71840/projects/show/46511758P)

**Synopsis**: From a WAFR, client would like us to evaluate and remediate security on the VPC (NACLs) and EC2 instances (Security Groups)

**Environment**: AWS EC2 security groups and VPC NACLs.

**Analysis**:

EC2 Instance: N. California Region, Instance ID: I-cdbb399, Name: RTR Web Server. Type: m1.large, Target type: m5.large, key-name: ocs-rtr. Two Volumes: Root (vol-f4e440a0) and D on Web (vol-4d14ba19)

VPC: vpc-f9e3d19d, NACL: acl-97db28f0, attached to two subnets

1. What NACLs and rules exist as built?
   1. Acl-97db28f0
      1. Inbound:
         1. 100 Allow Any/Any
      2. Outbound
         1. 100 Allow Any/Any
2. What security groups and rules exist as built?
   1. RTR-Web (sg-6993692d)
      1. Inbound:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Protocol | Port Range | Source | Description | NOTES |
| Custom TCP Rule | TCP (6) | 21 | 0.0.0.0/0 |  | FTP - is this needed? |
| HTTP (80) | TCP (6) | 80 | 0.0.0.0/0 |  | Yes – Web server |
| HTTPS (443) | TCP (6) | 443 | 0.0.0.0/0 |  | Yes – Web server |
| MS SQL (1433 | TCP (6) | 1433 | 0.0.0.0/0 |  | MS SQL Server is on the box, does it need a port open? Will be needed if RDS is utilized |
| RDP (3389) | TCP (6) | 3389 | 0.0.0.0/0 |  | THIS SHOULD BE LOCKED DOWN TO CLIENT CORPORATE |
| Custom TCP Rule | TCP (6) | 5000-5010 | 0.0.0.0/0 |  | Ephermal ports? |
| Custom TCP Rule | TCP (6) | 8090-8100 | 0.0.0.0/0 |  | Ephemeral ports? |
| Custom TCP Rule | TCP (6) | 27017 | 0.0.0.0/0 | Mongo | Is there a MondoDB instance somewhere?? |

**Resolution**:

1. If needed due to complexity, spin up a sandbox VPC using CloudFormation templates for testing. This will allow the changes to be reviewed and validated by the client
2. Recommended Changes:
   1. Acl-97db28f0

**Further Research**:

## Amazon Web Services

### Cloud Trail setup for Multiple A

### CloudTrail Logging

#### PROCEDURE: CloudTrail Logging to an S3 Bucket

* + - 1. *Summary*

This procedure sets up AWS CloudTrail logging from various AWS accounts to a common S3 Bucket in a single consolidation account.

* + - 1. *Reference*

https://aws.amazon.com/blogs/architecture/central-logging-in-multi-account-environments/

https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-receive-logs-from-multiple-accounts.html

* + - 1. *Input Variables*

The local Code Repository is here: D:\Users\brian.keith\projFTSLogging\Alexandria

The Branch in AWS Code Repository is Alexandria/projFTSLogging

* + - 1. *Validation*
      2. *Procedure*

Major Sections:

1. Create an Amazon S3 bucket for your central logging in the logging account

2. Create an AWS Lambda function log data transformation and decoding in the logging account

3. Create a central logging stack as a logging-account destination ready to receive streamed logs and deliver them to S3

4. Create a subscription in application accounts to deliver logs from a specific CloudWatch Log group to the logging account destination

5. Create Amazon Athena tables to query and analyze log data in your logging account

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | **Create an S3 Bucket** |  |
|  | Log into the AWS Console for the destination (Logging) account. Navigate in the console to the CloudFormation service |  |
|  | Using the file, cf\_projFTSLogging\_CreateS3Bucket\_template.json | The template will create a bucket, and have a 60 day archive to Glacier attached |
|  | Create a new stack, upload the source .json file  Click **Next** |  |
|  | Enter a stack name, and S3 Bucket name.  Click **Next** |  |
|  | Enter standard tags.  Click **Next** |  |
|  | Review the settings, Click **Create** |  |
|  | The CF template completes successfully; make note of the output |  |
|  |  |  |
|  | **CREATE the Data Transform Lambda Function** |  |
|  | Log into the AWS Console for the destination (Logging) account. Navigate in the console to the CloudFormation service |  |
|  | Use the file cf\_projFTSLogging\_CreateLambdaFunction\_template.json |  |
|  | Create a new stack, upload the source .json file  Click **Next** |  |
|  | Give the CF stack a name.  Click **Next** |  |
|  | Enter standard tags.  Click **Next** |  |
|  | Review the settings, Click the box acknowledge the IAM creation  Click **Create** |  |
|  | Upon success, check the output and make note of the ARN created for the Lambda function | arn:aws:lambda:us-west-2:930929710527:function:FTSprojCentralLoggingCrea-FirehoseDataProcessingFu-KN01S6XDBBI8 |
|  |  |  |
|  | **Create the Destination and Kinesis Streams** |  |
|  | Log into the AWS Console for the destination (Logging) account. Navigate in the console to the CloudFormation service |  |
|  | Use the file cf\_projFTSLogging\_CreateCentralLoggingDestination\_template.json |  |
|  | Create a new stack, upload the source .json file  Click **Next** |  |
|  | Enter a stack Name  Enter the Parameters – NOTE, the LogS3Location and SourceAccount will likely change based on destination  Click **Next** | Attempt # 2 |
|  | Enter standard tags.  Click **Next** |  |
|  | Review the settings, Click the box acknowledge the IAM creation  Click **Create** |  |
|  | Make note of the destination ARN in the outputs section |  |
|  | Parameters for FTS Prod – Flow Logs | arn:aws:logs:us-west-2:930929710527:destination:FTSCentralLog-Prod-Destination-Destination |
|  | **Create the log subscription in your application account** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

* + - 1. *Notes*

### Create a VPC Flow Log to CloudTrail

* + - 1. *Summary*

This procedure enables Flow Logs on a VPC and sends to CloudTrail Logging Group

* + - 1. *Reference*

https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs-cwl.html#flow-logs-cwl-create-flow-log

* + - 1. *Input Variables*
      2. *Validation*
      3. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Open the Amazon VPC Console at <https://console.aws.amazon.com/vpc> |  |
|  | In the navigation pane, choose Your VPCs or subnets |  |
|  | Select one or more VPCs or subnets and then choose Actions, Create flow log |  |
|  | For the Filter, specify the type of IP traffic data to log  Choose All, Rejected, Accepted |  |
|  | For Destination, choose Send to CloudWatch Logs |  |
|  | Destination Log Group type the name of the log Group |  |
|  | For IAM role, specify the name of the IAM role that has permissions to publish logs to CloudWatch logs |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Create a CloudTrail trail for use with Central Logging

* + - 1. *Summary*
      2. *Reference*
      3. *Input Variables*
      4. *Validation*
      5. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

*Notes*

## Citrix

## VMware

## Linode (Another Cloud Provider)

Client: CrimeDex

### Clone a Linode Instance

* + - 1. *Summary*

This procedure clones a Linode Instance (Virtual Machine). Useful for creating an instance for testing.

* + - 1. *Reference*

https://www.linode.com/docs/platform/disk-images/clone-your-linode/

* + - 1. *Input Variables*
      2. *Validation*
      3. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Log in to the Linode Manager |  |
|  | Select the Linodes you wish to clone, this will load it’s dashboard |  |
|  | Recommended: Click Shut down to power down the Linode |  |
|  | Click the Clone tabe to select the disks or configuration profiles you wish to clone |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

*Notes*

### MOP Reproduce a Linode Instance for CrimeDex

**Client: CrimeDex Linode Instance**

**Synopsis**: This MOP will provide automatic controls to restart stuck services and/or restart a host linode instance via a reboot. This is considered a stop gap measure, not a long term solution

**Environment**: Linode Instance – <https://manager.linode.com> (credentials in PasswordState

**Analysis**:

1. The original Linode Instance details
   1. Name: Production – Debian (Latest 32 bit (4.15.13-x86-linode125)
      1. VM Mode: Paravirtualization
      2. Boot: Latest 32 bit (4.15.13-x86-linode125)
      3. Default run level
      4. Memory (max 24576
      5. /d
   2. Disks
      1. /dev/sda Swap Image (4096 MB, swap)
      2. Debian 6 ProdTest Working (24576 MB, ext3)
      3. /dev/sdb Debian6-Production (364544MB, ext3)
   3. Other
      1. Root device – standard
      2. Distro Helper – yes
      3. Disable updated – yes
      4. Modules.dep Helper – yes
      5. Automount devtmpfs – yes
      6. Auto-configure Networking - not

**Resolution:**

Rough Draft:

1. Create a Clone or Restore from Backup

To restore from a backup, first create a new Linode instance. It must be large enough to accommodate the disk size. I chose the $80/month Large options

Time to restore from backup is about 15 minutes.

1. Discover Services
2. Install Monit
3. Configure and test Monit
4. Configure and test external Linode monitoring (Ping test?)
5. When validated, get client approval to touch production instance (roll back state?)
6. Install monit on production

**Further Research**

## Windows

## Storage

## Linux

## Network Devices

## Application

### Service or system overview

**Service or system name:**

### Business overview

What business need is met by this service or system? What expectations do we have about availability and performance?

*(e.g. Provides reliable automated reconciliation of logistics transactions from the previous 24 hours)*

### Technical overview

What kind of system is this? Web-connected order processing? Back-end batch system? Internal HTTP-based API? ETL control system?

*(e.g. Internal API for order reconciliation based on Ruby and RabbitMQ, deployed in Docker containers on Kubernetes)*

### Service Level Agreements (SLAs)

What explicit or implicit expectations are there from users or clients about the availability of the service or system?

*(e.g. Contractual 99.9% service availability outside of the 03:00-05:00 maintenance window)*

### Service owner

Which team owns and runs this service or system?

*(e.g. The Sneaky Sharks team (Bangalore) develops and runs this service:* [*sneaky.sharks@company.com*](https://mail.google.com/mail/?view=cm&fs=1&tf=1&to=sneaky.sharks@company.com) */ #sneaky-sharkson Slack / Extension 9265)*

### Contributing applications, daemons, services, middleware

Which distinct software applications, daemons, services, etc. make up the service or system? What external dependencies does it have?

*(e.g. Ruby app + RabbitMQ for source messages + PostgreSQL for reconciled transactions)*

### System characteristics

### Hours of operation

During what hours does the service or system actually need to operate? Can portions or features of the system be unavailable at times if needed?

#### Hours of operation - core features

*(e.g. 03:00-01:00 GMT+0)*

#### Hours of operation - secondary features

*(e.g. 07:00-23:00 GMT+0)*

### Data and processing flows

How and where does data flow through the system? What controls or triggers data flows?

*(e.g. mobile requests / scheduled batch jobs / inbound IoT sensor data )*

### Infrastructure and network design

What servers, containers, schedulers, devices, vLANs, firewalls, etc. are needed?

*(e.g. '10+ Ubuntu 14 VMs on AWS IaaS + 2 AWS Regions + 2 VPCs per Region + Route53')*

### Resilience, Fault Tolerance (FT) and High Availability (HA)

How is the system resilient to failure? What mechanisms for tolerating faults are implemented? How is the system/service made highly available?

*(e.g. 2 Active-Active data centres across two cities + two or more nodes at each layer)*

### Throttling and partial shutdown

How can the system be throttled or partially shut down e.g. to avoid flooding other dependent systems? Can the throughput be limited to (say) 100 requests per second? etc. What kind of connection back-off schemes are in place?

#### Throttling and partial shutdown - external requests

*(e.g. Commercial API gateway allows throttling control)*

#### Throttling and partial shutdown - internal components

*(e.g. Exponential backoff on all HTTP-based services + /health healthcheck endpoints on all services)*

### Expected traffic and load

Details of the expected throughput/traffic: call volumes, peak periods, quiet periods. What factors drive the load: bookings, page views, number of items in Basket, etc.)

*(e.g. Max: 1000 requests per second with 400 concurrent users - Friday @ 16:00 to Sunday @ 18:00, driven by likelihood of barbecue activity in the neighborhood)*

#### Hot or peak periods

\_

#### Warm periods

\_

#### Cool or quiet periods

\_

### Environmental differences

What are the main differences between Production/Live and other environments? What kinds of things might therefore not be tested in upstream environments?

*(e.g. Self-signed HTTPS certificates in Pre-Production - certificate expiry may not be detected properly in Production)*

### Tools

What tools are available to help operate the system?

*(e.g. Use the queue-cleardown.sh script to safely cleardown the processing queue nightly)*

### Required resources

What compute, storage, database, metrics, logging, and scaling resources are needed? What are the minimum and expected maximum sizes (in CPU cores, RAM, GB disk space, GBit/sec, etc.)?

### Required resources - compute

*(e.g. Min: 4 VMs with 2 vCPU each. Max: around 40 VMs)*

### Required resources - storage

*(e.g. Min: 10GB Azure blob storage. Max: around 500GB Azure blob storage)*

### Required resources - database

*(e.g. Min: 500GB Standard Tier RDS. Max: around 2TB Standard Tier RDS)*

### Required resources - metrics

*(e.g. Min: 100 metrics per node per minute. Max: around 6000 metrics per node per minute)*

### Required resources - logging

*(e.g. Min: 60 log lines per node per minute (100KB). Max: around 6000 log lines per node per minute (1MB))*

### Required resources - other

*(e.g. Min: 10 encryption requests per node per minute. Max: around 100 encryption requests per node per minute)*

### Security and access control

### Password and PII security

What kind of security is in place for passwords and Personally Identifiable Information (PII)? Are the passwords hashed with a strong hash function and salted?

*(e.g. Passwords are hashed with a 10-character salt and SHA265)*

### Ongoing security checks

How will the system be monitored for security issues?

*(e.g. The ABC tool scans for reported CVE issues and reports via the ABC dashboard)*

### System configuration

### Configuration management

How is configuration managed for the system?

*(e.g. CloudInit bootstraps the installation of Puppet - Puppet then drives all system and application level configuration except for the XYZ service which is configured via App.config files in Subversion)*

### Secrets management

How are configuration secrets managed?

*(e.g. Secrets are managed with Hashicorp Vault with 3 shards for the master key)*

### System backup and restore

### Backup requirements

Which parts of the system need to be backed up?

*(e.g. Only the CoreTransactions database in PostgreSQL and the Puppet master database need to be backed up)*

### Backup procedures

How does backup happen? Is service affected? Should the system be [partially] shut down first?

*(e.g. Backup happens from the read replica - live service is not affected)*

### Restore procedures

How does restore happen? Is service affected? Should the system be [partially] shut down first?

*(e.g. The Booking service must be switched off before Restore happens otherwise transactions will be lost)*

### Monitoring and alerting

### Log aggregation solution

What log aggregation & search solution will be used?

*(e.g. The system will use the existng in-house ELK cluster. 2000-6000 messages per minute expected at normal load levels)*

### Log message format

What kind of log message format will be used? Structured logging with JSON? log4j style single-line output?

*(e.g. Log messages will use log4j compatible single-line format with wrapped stack traces)*

### Events and error messages

What significant events, state transitions and error events may be logged?

*(e.g. IDs 1000-1999: Database events; IDs 2000-2999: message bus events; IDs 3000-3999: user-initiated action events; ...)*

### Metrics

What significant metrics will be generated?

*(e.g. Usual VM stats (CPU, disk, threads, etc.) + around 200 application technical metrics + around 400 user-level metrics)*

### Health checks

How is the health of dependencies (components and systems) assessed? How does the system report its own health?

#### Health of dependencies

*(e.g. Use /health HTTP endpoint for internal components that expose it. Other systems and external endpoints: typically HTTP 200 but some synthetic checks for some services)*

#### Health of service

*(e.g. Provide /health HTTP endpoint: 200 --> basic health, 500 --> bad configuration + /health/deps for checking dependencies)*

### Operational tasks

### Deployment

How is the software deployed? How does roll-back happen?

*(e.g. We use GoCD to coordinate deployments, triggering a Chef run pulling RPMs from the internal yum repo)*

### Batch processing

What kind of batch processing takes place?

*(e.g. Files are pushed via SFTP to the media server. The system processes up to 100 of these per hour on a cron schedule)*

### Power procedures

What needs to happen when machines are power-cycled?

*(e.g. \*\*\* WARNING: we have not investigated this scenario yet! \*\*\*)*

### Routine and sanity checks

What kind of checks need to happen on a regular basis?

*(e.g. All /health endpoints should be checked every 60secs plus the synthetic transaction checks run every 5 mins via Pingdom)*

### Troubleshooting

How should troubleshooting happen? What tools are available?

*(e.g. Use a combination of the /health endpoint checks and the abc-\*.sh scripts for diagnosing typical problems)*

### Maintenance tasks

### Patching

How should patches be deployed and tested?

#### Normal patch cycle

*(e.g. Use the standard OS patch test cycle together with deployment via Jenkins and Capistrano)*

#### Zero-day vulnerabilities

*(e.g. Use the early-warning notifications from UpGuard plus deployment via Jenkins and Capistrano)*

### Daylight-saving time changes

Is the software affected by daylight-saving time changes (both client and server)?

*(e.g. Server clocks all set to UTC+0. All date/time data converted to UTC with offset before processing)*

### Data cleardown

Which data needs to be cleared down? How often? Which tools or scripts control cleardown?

*(e.g. Use abc-cleardown.ps1 run nightly to clear down the document cache)*

## Log rotation

Is log rotation needed? How is it controlled?

*(e.g. The Windows Event Log ABC Service is set to a maximum size of 512MB)*

### Failover and Recovery procedures

What needs to happen when parts of the system are failed over to standby systems? What needs to during recovery?

### Failover

\_

### Recovery

\_

### Troubleshooting Failover and Recovery

What tools or scripts are available to troubleshoot failover and recovery operations?

*(e.g. Start with running SELECT state\_\_desc FROM sys.database\_\_mirroring\_\_endpoints on the PRIMARY node and then use the scripts in the db-failover Git repo)*

## Liquid Planner

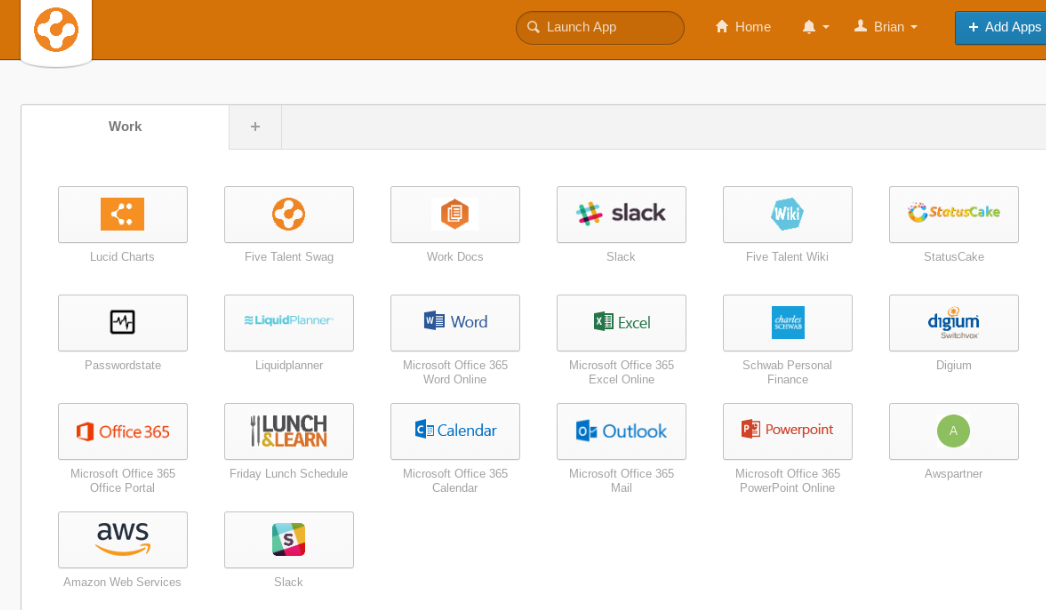
Project / Time Tracking Tool

https://app.liquidplanner.com/

## Okta

SSO solution

<https://fivetalent.okta.com/app/UserHome>



## Amazon Workdocs

This shared drive is accessible from multiple locations (sign ins) to store internal documents

Is this version controlled?

# Backup / Restore

## MOP – Create EBS Backup strategy using EBS Lifecycle Policies

**Client:**

\*\*\* Remediation: Implement Backup strategy for EC2's

<https://app.liquidplanner.com/space/71840/projects/show/46511796>

**Synopsis:** The client has several EC2 instances that need to be backed up. A simple strategy using the EBS Lifecycle policy (Not available as of July 25 in N. California) will create and retain several EBS Snapshots. Tagging is required for the source volumes. Frequency can be either every 12 hours, or every 24 hours.

**Environment:** The instance contain several volumes each. Need to validate with the client the RPO/RTO time ranges, and number of backups to keep.

**Analysis:**

**Resolution: (rough draft example)**

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | Create a new Instance. (OR add tags to an existing instance) |  |
|  | In the AWS Management Console,drill down to EC2 >> EBS >> LifeCycle Policy.  Click create SnapShot Lifecycle Policy |  |
|  | Enter Values  Description: Real Time Research Test Policy  Select TAG: EBSBackupFrequency:12Hour  Retain: 4 copies |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

**Further Research:**

<https://aws.amazon.com/answers/account-management/aws-tagging-strategies/>

<https://aws.amazon.com/blogs/aws/new-lifecycle-management-for-amazon-ebs-snapshots/>

# Antivirus and Related Security

# Systems Provisioning

## Create Workstation for TriComm

Workspace for Tricomm

AMI: workspace-jed

Size: t2.large

security groups:fivetalent ips sg-a703b1defivetalent

IAM ROle: EC2WorspaceROle

keypair: tricon-workspace

Name: andrew dunaway

default username: jed / Abc123!

Add a new user (alex)

Change computer name to username AlexGarten

Need to get users IP address and add to FiveTalent security group

## Create a new single instance Debian WordPress Server

* + - 1. *Summary*

This procedure walks through the steps to create a LAMP stack on Debian, and then install the WordPress application. This is useful for testing as front-end webserver.

Note: ssh to Debian instances as user **admin** using your SSH key, and then **sudo -I** to gain root access.

* + - 1. *Reference*

<https://linuxconfig.org/how-to-install-wordpress-on-debian-9-stretch-linux>

* + - 1. *Input Variables*

Assumption: A base EC2 Instance running Debian 9 is already running. It has access to the internet.

* + - 1. *Validation*
      2. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  | SSH To the Target Server. Become root |  |
|  | Install MariaDB(MySQL  #apt install mariadb-client mariadb-server |  |
|  | Install php  #apt install php7.0 php7.0-mysql |  |
|  | Install Apache  #apt install apache2 libapache2-mod-php7.0 |  |
|  | Create file /var/www/html/index.php | <?php phpinfor(); ?> |
|  | Install phpMyAdmin  #apt install phpmyadmin |  |
|  | Browse to the public IP to verify website is working |  |
|  | WORDPRESS |  |
|  | Create the Database  #mysql – u root -p  MariaDB [(non)]> CREATE USER ‘wpuser’@”localhost’ IDENTIFIED BY ‘userpassword’;  CREATE DATABASE wp\_database;  GRANT ALL ON ‘wp\_database’.\* TO ‘wpuser’@’localhost’;  FLUSH PRIVILEGES;  exit |  |
|  | $cd Downloads  $ wget <https://wordpress.org/latest.tar.gz>  $tar xpf latest.tar.gq |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

* + - 1. *Notes*

# User Provesioning to AWS

# Key Contact Information

# AWS IAM Users, Groups, Roles

# Appendix

## Customer Processes to be known

## Other environment-specific information for reference

### AWS Partner Portal

## Index of Terms

## Section Templates

### Server Template

### Device Grid

* + - 1. *General Information*

<<Device Description>>

* + - 1. *Technical Information*

|  |  |
| --- | --- |
| Server Name | SERVER ID |
| Key Running Services |  |
| Application Versions |  |
| OS Platform and Patch Level |  |
| Virtual Server Y/N | Yes |
| Hardware Platform |  |
| Domain |  |
| Location |  |
| CPU/Memory Allocation |  |
| Network Information | Adapters:  Em0 (00:00:00:00:00:00)  VLAN  **IP ADDR: x.x.x.x**  Subnet: 255.255.255.0  Gateway:  DNS: |
| Priority | Critical/Important/Low/Dev/Test |
| Disk/Share Information | Disks: |
| Additional Information | Primary Contact:  Application Contact: |
| Miscellaneous Notes |  |

* + - 1. *Configuration Notes*

Enumerate device specific notes, especially anything that is nonstandard.

* + - 1. *Reference Guides*

Paste links to online (internal/external) reference guides

### Procedure Template

* + - 1. *Summary*
      2. *Reference*
      3. *Input Variables*
      4. *Validation*
      5. *Procedure*

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

* + - 1. *Notes*
  1. NAT Rule Template
     + - 1. *Rule Template*

|  |  |  |  |
| --- | --- | --- | --- |
| Rule Name |  | Type |  |
| Original IP : Port |  | **Translated IP : Port** |  |
| Protocol |  | **Enabled** |  |
| Description |  | **Notes** |  |

### WIKI Template

**Client**

**Synopsis**

**Environment**

**Analysis**

**Resolution**

|  |  |  |
| --- | --- | --- |
| Step | Process | Screen Shot |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Procedure Complete. |  |

**Further Research**

# Document Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Revision Number | Revision Date | Section Reference | Summary of Changes | Author | Reviewer |
| 1.0 | July 10, 2018 | Entire Doc | Setup structure and headings, TOC | Brian Keith |  |
| 1.2 | July 25, 2018 | Section 8.1 Tagging Categories | Listing of Basic Tag categories | Brian Keith |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Headings Check

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

Heading 7

Heading 8

Heading 9

# Headings Check