Developing Infrastructure for Using Guacamole

Mark Lidd

mark.lidd@plus3it.com
New Capability: Developer Desktop

- Preconfigured Virtual Desktop for Software Development
  - No need to take time to setup developer environment
    - Software development applications are already loaded
    - JetBrains IDE, Eclipse, GitHub/GitLab, Visual Studio, AI Plugins

- Accessible from a Browser
  - No special client downloads

- Entirely OpenSource
  - No restrictions due to licensing

- Multiple O/S workstations available
  - Centos7 initially; Other Linux & Windows later

- Platform
  - Instance-based initially
  - Container-based (AKA Kubernetes) later for applications not requiring a full O/S

- Developers find same/similar environment on various projects/networks
Simple Concept: Virtual Developer Workstation

Requirements

- Centos7 based virtual workstation
  - Accessible from available windows/linux clients
    - Use RDP wrapped in TLS
  - Preconfigured with “typical” developer apps
    - Eclipse, compilers, editors, Docker, et. al
  - Meets security requirements

- Multi-Cloud “capable”

- Give developers “something”
  - That is easy to use - preconfigured
  - Saves them time from having to configure a workstation themselves
  - Reasonably secure
    - But doesn’t stop them from configuring their unique development environment

- Scale to hundreds of users

Solution

- Build a centos7 Instance
  - Preconfigured w/things a “developer” needs
    - Eclipse, Editors, Desktop
    - Development environment (Java, C++, etc.
    - Use Ansible to configure workstation
  - Set of tools preconfigured to meet security requirements
  - Save developer configuration time

- Access from a web browser
  - Use HTML5 (Most Web clients)
  - Apache Guacamole
  - Considered RDP client with TLS tunnel, but didn’t meet audit requirements
Really want to avoid

OK. I’ve got this great Open Source product. I’ve tested it and it works great. What do I do to provide this awesome functionality to my hundreds of users.

• All too typical:
  o Load onto an accessible instance
  o Update instance size when users complain of slowness
  o Reboot if problems
  o Spend inordinate amount of time with users; changing passwords, rebooting, general support, updating public/private keys, etc.
  o “Been there; Done that”
  o Security is an afterthought
Talk’s Focus

- **Guacamole Server (Docker Container)**
- **Developer Desktop**

Proof-of-Concept

- **Load balancer**
- **Firewall**
- **Auto scaler**

Production

- **Maintainable**: secure, key rotation, updates, new versions
- **AWS focused now but vendor neutral growth**
- **Load balancer** does a health check and if failure then **Autoscale** will in turn fail and regenerate the instance
- **AWS RDS** will automatically backup and update the Postgres database
Use a Commercial Product? Possible But…

• Typical Issues
  o Workstation not general enough of customization not sufficient
  o Not secure enough
  o Needs custom client download
  o License fees
  o Lack of control
  o Not multi-cloud
  o Proprietary

• Our Needs
  o We want to create a paradigm to support ease of use for the developer
  o Want flexibility
  o Not force a paradigm on the developer
Guacamole:
Aside from a really yummy dip for chips...

- Guacamole is:
  - A protocol
  - A Clientless Remote Desktop Gateway
    - Clientless: Only a web browser required - no plugins, no software installs
    - Remote Desktop: Support for common remote desktop protocols
      - Kubernetes
      - RDP
      - SSH
      - Telnet
      - VNC
  - Gateway: Web-based, authentication, and permission control

- Started by Michael Jumper and James Muehlner, circa 2010
- Initially developed on SourceForge
- Entered Apache Software Foundation (ASF) “Incubator” project in 2016
- Graduated to ASF Top Level Project (TLP) in late 2017
- Licensed under the Apache 2.0 License

Slide contents provided by Nick Couchman/Apache Guacamole
What “Production” means (to me)

OpenSource project provides function “Buffet” to meet production needs (OAuth, OpenID, SAML, etc.)

- **Configuration**: Understand configuration options
- **Traceability**
  - Code
  - Glue code
  - Logging
- **Cost Control**
- **Maintainability**
  - Public/private keys; UserIDs/passwords
  - Administration
    - Infrastructure
    - Users
    - Support
- **OpenSource Software is “Ready to Use”**
  - All project CM done by the opensource project
  - Only configuration file(s) need to be generated

- **Add “stuff” not supplied by the project repo**
  - Site Specific Information & Localization
    - Branding
    - Security needs
    - Infrastructure
- **Other Considerations**
  - Updates
    - Project software as needed
    - Other software
      - Scripts
      - Downloaded applications
  - Security in general
  - Keep project software load “clean”; no changes
  - UserIDs/passwords, public/private keys secure

---

 Community Over Code 2023
Utilize AWS for Production Environment Integrations

• Log File Histories:
  o AWS Cloudwatch & AWS S3

• Statistics:
  o AWS Cloudwatch

• Alerting (Alarms & Notifications)
  o AWS Cloudwatch

• Automation: Building & Management
  o CloudFormation
  o AutoScaling
  o Load Balancer
  o Other AWS Infrastructure: Security groups, IAM Policy, VPC/subnets, AMI, instances

• Secrets
  o AWS Secrets
Additional Needs for Production (1/3)

• Guacamole needs a datastore to store user & connection data
  o Default is a data file for connection parameters and UserIDs/passwords
    • Other methods supported: we use AWS RDS (Postgres) and OpenID

• Issues Management:
  o Git
  o Software Development Management: Jira

• Manage & store public/private keys:
  o Update & Rotate Keys
  o Includes signed public keys not generally in keystores
  o Key Stored in AWS Secrets
    • Managed access via AWS IAM policies
Additional Needs for Production (2/3)

• Congestion Handling
  o AWS Autoscaling
  o AWS Load Balancer

• Create base AMIs
  o Currently: run scripts manually
  o Planned: Create monthly via AWS Lambda

• Configuration Management
  o “Vanilla” repository
    • Scripts, templates, infrastructure (Cloudformation)
  o Site Specific Repository
    • Location Unique identifiers repository (yaml/json files)

• Blob Repository
  o Nexus
Additional Needs for Production (3/4)

• Location Unique Credentials:
  o AWS Secrets
  o Scripts to generate/access these Credentials
  o Infrastructure parameters (VPC, Subnet, AMI IDs, IAM Policy. etc.)

• Log File Histories; Statistics; Alerting (Alarms & Notifications)
  o AWS Cloudwatch & AWS S3
  o Use the unified CloudWatch agent

• Automation
  o Building
    • Should be scripted; parameters in yaml file
  o Management
Additional Needs for Production (4/4)

• General Security

• Architecture:
  - Isolated with security groups in separate zones/subnets
    • Developer Desktop instances
    • Guacamole & Database
    • Load balancer (Accepts only HTTPS (port 443))

• Isolation via subnets; AWS security groups

• Encrypted data at rest

• Encryption in transit

• AWS SMS
  - Periodic updates; Periodic security scans

TLS Encryption:
Guacamole and Guacd
User and load balancer
Load balancer and Guacamole
Guacd and User’s Desktop Instance
Guacamole and database
All AWS services connections
Feature Build Breakdown

• Build custom AMIs
  o Create DD AMI w/ needed software for Developer Desktop Instance
  o Create Guacamole AMI w/Guacamole & Guacd
  o Use Hashi's Packer w/ Ansible
  o Template file driven python scripts
    • For DD EC2 deployment
    • For CloudFormation Guacamole deployment
Secrets & Management

Secrets

- 6 public/private key pairs to rotate (for TLS)
  - Load Balancer <-> user
  - Tomcat <-> Load Balancer
  - Servlet (guacamole.war) <-> Guacd
  - Servlet (guacamole.war) <-> Database (SQL)
  - Guacd <-> User’s Instance (RDP)
  - OpenID server <-> Servlet (Guacamole.war)

- Database UserID/password

- Other tokens/public keys
  - Nexus Blob Repository token (needed during AMI build)
  - AWS public keys

- User Login to Guacamole managed with OpenID server

- Desktop User Logins/Passwords also stored by Guacamole

Secrets Management

- Scripts to push keys into AWS Secrets
  - Run as needed
  - Can create self-signed or use signed keys as needed

- Public keys added to keystore during AMI build (Packer/Ansible)

- Private keys read from AWS Secrets using Cloud-Init during instance deploy

- Refresh keys done by
  - Pushing keys to AWS Secrets
  - Rebuilding AMI
    - w/public keys
  - Re-deploying using AWS Autoscaler
    - private keys added
Encryption Key Updates

• Public key updates need new AMI
  o Update keys in AWS Secrets
  o Rebuild AMI

• Private key updates need new deploy
  o Update keys in AWS Secrets
  o Redeploy (rolling update)
    • Allows existing connections to drain
    • New connections are routed to new instance

Outage; debug; fix keys identified
Create/download new key pair
Update
Repeat for each key
Keys/Passwords Scripts: Calls & Returns

• Wrote script to either read keys from files or generate a self-signed key pairs and load keypairs into AWS Secrets

• ./gen-guac-ssk
  o -c <true if create self-signed keys | false to read keys from files>
  o -p < true to push keys and passwords to AWS Secrets manager | false to export keys and passwords to env>

• To rotate keys:
  o Run: source ./gen-guac-ssk -c true -p true
  o Then use AWS Autoscaling to terminate current Guacamole instance and deploy new
AMI Generation

• Packer scripts with Ansible plugin
  o Site specific variables defined in Packer HCL yaml file
  o Site specific variables defined in Ansible Vars directory yaml file
  o Parameters (source AMI Id, et. al.) in bash wrapper script

• Separate but similar scripts for Developer Desktop AMI & Guacamole (Guacamole & Guad) AMI
AMI Scripts: Calls & Returns

- **Developer Desktop AMI**
  ```bash
  #!/usr/bin/bash
  if [ "x$1" == "x" ]; then echo "var file not present - exiting"; exit 1; fi
  echo "using "$1" as packer var file"
  read -p "Are you sure? " n 1
  if [[ ! $REPLY =~ ^[Yy]$ ]]; then echo "response must be "Y" or "y" to continue. Exiting"; exit 2; fi
  echo
  # create new version number
  MAJOR="0"
  MINOR=$(($(cat DD_version)+1))
  echo $MINOR > DD_version
  PAD=""
  if [[ MINOR -lt 10 ]]; then PAD="0"; fi
  if [[ MINOR -lt 100 ]]; then PAD="0$PAD"; fi
  DD_version="V${MAJOR}.${PAD}${MINOR}"
  echo "NEW AMI Version: $DD_version"
  # Run Pack
  # packer validate -var "ami_force_deregister=true" ami
  # get certs for DD nexus repo
  ./get_certs
  export DD_ADMIN='DEFAULT_USER'
  packer build -var-file "$1" -var "dev_wkstn_version=DD_version" ami.dd
  ```

  Note: var file contains files of variables in YAML format
  YAML files provide configuration management

- **Guacamole AMI**
  ```bash
  #!/usr/bin/bash
  if [ "x$1" == "x" ]; then echo "var file not present - exiting"; exit 1; fi
  echo "using "$1" as packer var file"
  read -p "Are you sure? " n 1
  if [[ ! $REPLY =~ ^[Yy]$ ]]; then echo "response must be "Y" or "y" to continue. Exiting"; exit 2; fi
  echo
  # create new version number
  MAJOR="0"
  MINOR=$(($(cat Guac_ami_version)+1))
  echo $MINOR > Guac_ami_version
  PAD=""
  if [[ MINOR -lt 10 ]]; then PAD="0"; fi
  if [[ MINOR -lt 100 ]]; then PAD="0$PAD"; fi
  Guac_ami_version="V${MAJOR}.${PAD}${MINOR}"
  echo "NEW Guacamole AMI Version: $Guac_ami_version"
  # Run Pack
  # packer validate -var "ami_force_deregister=true" ami
  # get certs for DD nexus repo
  pushd .. && ./get_certs && popd
  export DD_ADMIN='DEFAULT_USER'
  packer build -var-file "$1" -var "guac_server_version=$Guac_ami_version" ami.guac
  ```
User Desktop Deployment

• Bash script with call to generalized Python script EC2 deployment
  o Generates the User’s instance thru calls to EC2 API (BOTO3)
  o Updates Guacamole database
    • Adds user data
    • Adds associated instance connection data
  o EC2 Deployment is template driven

• Details:
  o Script arguments include instance UserID, Password, public key
  o Reads a template yaml file containing subnet ID, AMI ID, userdata, etc.
  o Creates yaml file with all particulars for visual verification and CM; also logs deploy data for historical record.
User Desktop Deployment: Calls

- A wrapper script for a Python program that deploys an EC2 instance
  - Creates a user’s DD instance and updates the guacamole database:
    - `./dd-deploy -u user -p password -k "ssh-rsa AAAA...." -a "ami-0000000000000"

- YAML files provide configuration management

- Example parameter file for runec2:
  ```
  REGION : 'us-east-1'
  EC2-OPERATION : 'create'
  
  EC2-PARAMETERS :
  - 'SecurityGroupIds' :
    - 'sg-00000000'
  - 'SubnetId' : 'subnet-0000000000000'
  - 'InstanceType' : 't3.medium'
  - 'KeyName' : 'my-pem'
  - 'ImageId' :
  - 'MaxCount' : 1
  - 'MinCount' : 1
  - 'TagSpecifications' :
    - 'ResourceType' : 'instance'
      - 'Key' : 'Name'
        - 'Value' : 'DD'
      - 'Key' : 'ID'
        - 'Value' : '00001'
  - 'UserData' :
    - '#!/usr/bin/bash
    - #
    - #setup named user
  ```
Guacamole Capacity

- AWS Load balancer scales as needed to handle users volume
- Autoscaling used to create more guacamole instances as needed
  - Average CPU usage > 60% triggers autoscaling up
Guacamole Deployment

• Bash script with call to generalized Python script Cloudformation deployment
  o Specifies the Guacamole instance thru autoscale template init and userdata Instance
  o Setup up statistics collection, alarms and TBD
  o Parameter driven using yaml/json file to provide site parameters
    • VPC ID, subnet ID, AMI ID, autoscale parameters, IAM policy ID

• Details:
  o Script arguments include CloudFormation update, create, delete.
  o Reads a parameter file containing VPC ID, subnet ID, AMI ID, autoscale parameters, IAM policy ID, etc.
  o Produces log file for record
Guacamole Deploy: Calls & Returns

- Deploy cloudformation template:
  
  python runstk.py --log-level DEBUG -pf parameters-guac-server.yaml

  Sample parameters-guac-server.yaml:

  Region : 'us-east-1'
  StackName : 'My_Stack'
  DisableRollback : 'True'
  CAPABILITIES : 'CAPABILITY_IAM'
  TEMPLATE-FILE : 'my_cfn.yaml'
  TEMPLATE-OPERATION : 'create'
  STACK-PARAMETERS :
  - VPCId : 'vpc-000000000'
  - VPCIdCidr : "10.10.10.0/24"
  - SubnetID : "subnet-0000000000000"
  - SecurityGroups : 'sg-0000000000000, sg-0000000000001, sg-0000000000002'
  - SystemsManagerAccess : 'false'
  - KeyName : "my-pem"
  - RedirectURL : 'internal-dev-00000000000.us-east-1.elb.amazonaws.com'
  - RedirectURLPort : ""
  - InstanceType : "t3.med"
Planned

• Currently all scripts return error codes upon exit. But more is needed
  • Functionality tests needed to determine validity
    • DD AMI; Guacamole AMI
    • DD instance after deployment
    • Guacamole instance after deployment
      • Currently LB does a simple HTTPS ping but doesn’t test functionality
      • Test Guacamole/Tomcat functionality
      • Test Guacd functionality

• Other Developer instances
  • Other versions of Linux instances; windows versions
  • Other versions of K8s O/S instances
Next Generation Architecture
Thank You
Questions?