AWS Devops Notes

Service Catalogs – create stacks backed by CF through a portal
CloudFormation circular dependency – two resources dependent on each other or resource dependent on self
CodeDeploy deployment groups?
Kinesis
EC2 Inspector, Trusted Advisor, Config, GuardDuty, Macie
EC2 scheduled events
DynamoDB GSI and LSI, projected attribute, sort key
CW Event targets
Storage Gateway

Bonso #1

1 – You can configure rules in CW Events to alert you to changes in OpsWorks Stacks w/ 'detail.initiated_by' set to 'auto-healing'; associate w/SNS topic

2 – Use CW Logs agent for on prem/AWS resources to collect logs, store in S3 in central account, set up S3 trigger that invokes a Lambda to analyze logs to detect irregularities, analyze logs using Athena; retrieve custom metrics using **StatsD** and **collectd** protocols

3 – To ensure all EBS volumes are encrypted across all accounts using AWS Organizations w/detailed report – set up Config rule w/Lambda on all target accounts, collect data from accounts/regions using Config aggregators, export report to S3 then deliver notifications using SNS

4 – When you modify the db engine for your db instance in a multi-az deployment, RDS upgrades both primary/secondary instances at same time – in this case the db engine for the entire multi-az deployment is shut down during the upgrade – so in AWS::RDS::DBInstance resource type update the **EngineVersion** to latest version – create a second application stack and launch a new Read Replica with same properties as primary db instance that will be upgraded – finally perform **Update Stack**

5 – After the last deploy action of the pipeline, set up a manual approval action and inform the team of the stage being triggered using SNS – in CodeBuild add the required actions to automatically do unit/int tests – add a deploy action to deploy the app to the next stage at the end of the pipeline

6 – CodeCommit feature branch for features

7 – **hooks** section of AppSpec for CodeDeploy can access environment variables; you can develop a custom shell script using **DEPLOYMENT_GROUP_NAME** in CD to identify which deployment group – in appspec file add a reference to this script as part of the BeforeInstall hook – configure log level settings of the instance based on the result of the script

APPLICATION_NAME - The name of the application in CodeDeploy that is part of the current deployment *(for example, WordPress_App)*.

DEPLOYMENT_ID - The ID CodeDeploy has assigned to the current deployment *(for example, d-AB1CDEF23)*.

DEPLOYMENT_GROUP_NAME - The name of the deployment group in CodeDeploy that is part of the current deployment *(for example, WordPress_DepGroup)*.

DEPLOYMENT_GROUP_ID - The ID of the deployment group in CodeDeploy that is part of the current deployment *(for example, b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE)*.

LIFECYCLE_EVENT - The name of the current deployment lifecycle event (for example, AfterInstall).

These environment variables are local to each deployment lifecycle event.

The following script changes the listening port on an Apache HTTP server to 9090 instead of 80 if the value of **DEPLOYMENT_GROUP_NAME** is equal to `Staging`. This script must be invoked during the `BeforeInstall` deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
 sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

8 – ALB ASG EC2 instances deployed via CICD – track health and latency of services, halt services if threshold is reached – calculate avg latency using CloudWatch Metrics – associate CW Alarm with CodeDeploy group – when latency increases beyond threshold it will trigger alarm automatically stopping deployment

9 – you can use Lambda to validate part of deployment of CodeDeploy for an ECS app – specify hook **AfterAllowTestTraffic** to validate test traffic before deploying to prod

10 – By setting up MinSize and MaxSize params of Auto Scaling Group to 1 you can ensure your EC2 instances can recover again in the event of system failure with exactly the same params defined in CloudFormation template – this strategy provides high availability with least possible cost
Set up a CloudFormation child stack template which launches an Auto Scaling group consisting of just one EC2 instance then provide a list of ENIs, hostnames and the specific AZs as stack parameters. Set both the **MinSize** and **MaxSize** parameters of the Auto Scaling group to 1. Add a user data script that will attach an ENI to the instance once launched. Use CloudFormation nested stacks to provision a total of 10 nodes needed for the cluster, and deploy the stack using a master template
11 – ASG of EC2 instances in VPC unsuccessful calls to api – look for **REJECT** records **in VPC flow logs** originating from ASG – verify **egress** security group rules of ASG allow outgoing traffic to external API
12 – track who is making S3 object changes using **data events** in CloudTrail to track any object changes – create a rule to run your Lambda in response to S3 data event that checks if the user who initiated the change is an admin; Data events are disabled by default
13 – Custom resources enable you to write custom logic in CloudFormation templates – by using custom resources and Lambda you can get IDs of latest AMIs and reference it in the launch configuration resource block
14 – Automate patching for hybrid architecture:

1. Set up IAM service role for Systems Manager to enable the service to execute STS AssumeRole operation; allow the generation of service tokens by registering the IAM role; use the role to perform the managed instance activation
2. Download and install the SSM Agent on the hybrid servers by using the activation codes and ids you obtained; register the servers or vms on prem to Systems Manager service; in SSM console, the hybrid instances will show with an 'mi-' prefix; apply patching using Patch Manager

15 – Use **Personal Health Dashboard** and CloudWatch Events to track AWS initiated activities to your resources; create an event using CW Events which can invoke a Lambda to send notifications to company's Slack channel
16 – Launch an ALB and use a blue/green deployment for releasing new application versions. Associate the ASG and the ALB target group with the deployment group. Create a custom deployment configuration for the deployment group in CodeDeploy with minimum healthy hosts defined as 50% and configure it to also terminate the original instances in the Auto Scaling group after deployment. Use the BeforeAllowTraffic Traffic hook within appsec.yml to purge the temporary files.
17 – Notifications for Trusted Advisor as frequently as possible:

1. Lambda runs daily to refresh TA via API and publish a message to SNS topic to notify subscribers
2. Lambda runs daily to refresh TA via API and send results to CW Logs; metric to send alarms
3. CW Events to monitor TA checks and set trigger to send email using SNS

18 – Multi region disaster recovery plan w/LOWEST recovery time and LEAST data loss:
Launch stack in another region using CF template; create RDS Read Replica in other region then enable cross-region replication between original S3 bucket and new one; promote RDS Read Replica as master in event of failover; increase capacity of ASG to improve scalability
NOTE: You can NOT host a standby db in another region

	DISASTER RECOVERY		COST	SCOPE
	RTO	RPO		
AUTOMATED BACKUPS	GOOD	BETTER	LOW	SINGLE REGION
MANUAL SNAPSHOTS	BETTER	GOOD	MEDIUM	CROSS-REGION
READ REPLICAS	BEST	BEST	HIGH	CROSS-REGION

19 – Remediate CloudTrail from being disabled on accounts automatically:
Use **cloudtrail-enabled** Config managed ruled w/periodic interval of 1 hour to evaluate whether your account enabled CT; set up a CW Events rule for Config rules compliance change; launch Lambda that uses the SDK and add the ARN of the Lambda as the target in CW Events rule; once a **StopLogging** event is detected, the Lambda will re-enable the logging for that trail by calling **StartLogging**
NOTE: cloudtrail-enabled doesn't not automatically remediate disablement and is only available for periodic
20 – CF stack deletion fails for S3 bucket that still has contents – to fix, modify Lambda of custom resource to recursively empty the bucket

21 – CodeDeploy Lambda: Canary (2 increments), Linear (equal increments), All-at-once
22 – ECS app, push new image to registry, stop and start all tasks to ensure containers have latest version, however tasks occasionally run old image – **restart ECS agent**
To have your service use a newly updated Docker image with the same tag as in the existing task definition (for example, `my_image:latest`) or keep the current settings for your service, select Force new deployment. The new tasks launched by the deployment pull the current image/tag combination from your repository when they start.
The Force new deployment option is also used when updating a Fargate task to use a more current platform version when you specify `LATEST`. For example, if you specified `LATEST` and your running tasks are using the `1.0.0` platform version and you want them to relaunch using a newer platform version.
23 – ECS, ALB – enable access logging on ALB and create IAM policy and attach it to ecsInstanceRole – install CW Logs agent on ECS instance – use **awslogs** Logs Driver in ECS task definition; create CW subscription filter w/Kinesis to analyze logs; export both to S3
24 – If you plan to launch an ASG of EC2 instances, you can configure **AWS::AutoScaling::AutoScalingGroup** in your CloudFormation template. To configure EC2 instances launched as part of the group, you can specify a launch template. You can add an **UpdatePolicy** attribute to your ASG to perform rolling updates when a change has been made to the group. To specify how CF handles replacement updates for an ASG use the **AutoScalingReplacingUpdate** policy. This policy specifies whether CF replaces an ASG w/ a new one or replaces only the instances in the ASG. During replacement, CF retains the old group until it finishes creating the new one. If the update fails, CF can roll back to the old ASG. While CF creates the new group, it doesn't detach or attach any instances. After successfully creating the new ASG, CF deletes the old ASG. When you set the **WillReplace** param, remember to specify a matching CreationPolicy. If the min number of instances don't signal success w/in the Timeout period, the replacement update fails and CF rolls back to the old ASG.
If you set WillReplace to false, then AutoScalingRollingUpdate will take precedence over AutoScalingReplacingUpdate. The rolling update doesn't maintain the total number of active EC2 instances during deployment.

```
UpdatePolicy:
  AutoScalingReplacingUpdate:
    WillReplace: true
```

25 – db creds can be stored in Secrets Manager – make sure IAM role for EC2 instances using db have access to SM
26 – Lambdas deployed via CodeDeploy w/Lambdas using validation scripts – need to test all APIs before fully deploying to prod, check APIs during deployment and be notified of errors as wells as auto rollback

1. Configure Lambda validation scripts to run during deployment and CW Alarm to trigger rollback failure
2. Associate CW Alarm to deployment group to send notification to SNS when 5xx threshold reached
3. Define Lambda validation scripts on AppSpec lifecycle hook during deployments to run validation using test traffic and trigger rollback if check fails

27 – Maintain AWSCodeCommitPowerUser managed policy but create additional policy to include a Deny rule for codecommit: GitPush – add restriction for specific repo in resource statement w/condition for master branch
AWSCodeCommitFullAccess, AWSCodeCommitPowerUser (can't delete repos), AWSCodeCommitReadOnly
You can NOT modify AWS managed policies

28 – Keep db creds in Secrets Manager and encrypt w/KMS – set up IAM Role for ECS task execution role and reference it w/your task definition which allows both KMS and Secrets Manager – w/in your container definition

specify secrets w/name of env variable to set in the container and the full ARN of Secrets Manager secret to present to the container

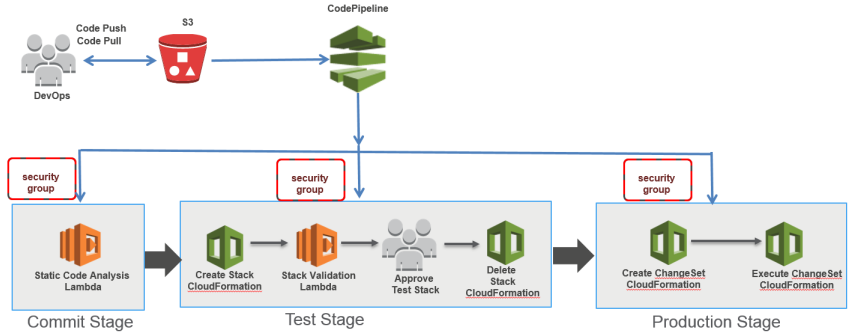
29 – Improve RTO and RPO of website missing 3 hrs of data for cross region failover:

Use Step Functions w/2 Lambdas that call RDS API to create a snapshot of the db, create a cross-region snapshot copy, and restore the db instance from a snapshot in the backup region. Use CW Events to trigger the function to take a db snapshot every hour. Set up an SNS topic that will receive published messages from AWS Health API, RDS availability and other events that will trigger the Lambda to create a cross-region snapshot copy. During failover, configure the Lambda to restore the db from a snapshot in the backup region.

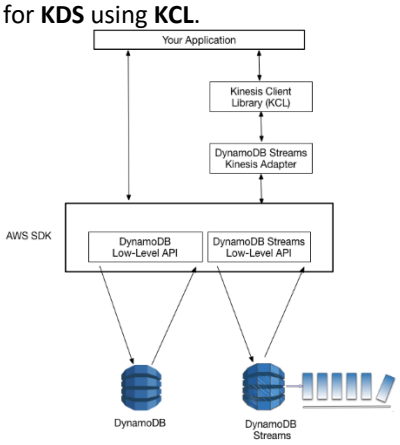
When you copy a snapshot to a different region, the first copy is a full snapshot while the next are incremental.

30 – In CodePipeline stage, you can specify parameter overrides for CloudFormation actions. **Parameter overrides** let you specify template parameter values that override values in a template configuration file.

To specify config variables and static attributes for each environment launch a pipeline that has multiple stages for each env and configure it to use input params. Switch the associated UserData of the EC2 instances to match the env where the app stack is being launched using CF **mappings**. Specify parameter overrides for CF actions.



31 – Using **Kinesis Adapter** is the recommended way to consume streams from **DynamoDB**. You can write apps for **KDS** using **KCL**.



32 – Set up a CW Events rule to monitor CodeDeploy w/Lambda as target. Configure the frule to send out a message to the Slack channel if the deployment fails. Configure CodeDeploy to use the Roll back when a deployment fails setting.

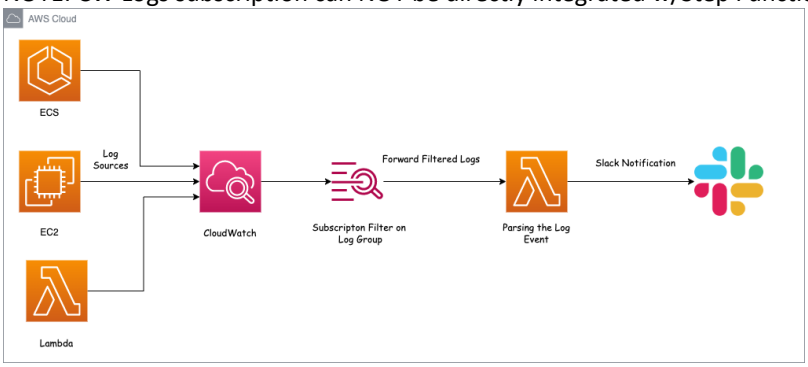
33 – Configure CodeBuild project to use IAM Role w/required permissions and remove AWS creds from buildspec.yml. Run scp and ssh commands using Systems Manager Run Command.

Use Systems Manager Param Store to create a db password secure string.

34 – Use an active-passive failover configuration when you want a primary resource/group to be available the majority of the time and you want a secondary resource/group to be on standby in case all primary resouces become unavailable.

1. Specify a **Failover** routing policy. Set up alias records in Route 53 that route traffic to AWS resources. Set the **Evaluate Target Health option to Yes** then create all of the required non-alias records.
2. Configure health checks that monitor an endpoint you specify either by IP or domain name. Configure the nacl and route table to allow Route 53 to send requests to the endpoints specified in health checks.

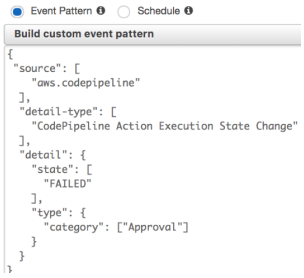
35 – Config provides a visual dashboard to help quick spot non compliant, misconfigured resources. You can store the data in S3 and use **QuickSight** to analyze the dataset.
36 – When an instance is paused it remains in a wait statue until you **complete the lifecycle action.**
You can prepare a CF template using ASG to launch EC2 instances w/ an associated lifecycle hook for Instance Terminate. Store the 100 license codes to a DDB table. Pull an available license from DDB using **User Data** script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated.
37 – EC2, ASG, ALB, Route53 – geo isolated DR site w/RPO 5min and RTO 2 hrs:
Clone app stack except RDS in a different Region. Create Read Replica in the new region and configure the new app stack to point to the local RDS db instance. Set up a failover routing policy in Route 53 that will automatically route traffic to the new app stack in the event of an outage.
38 – Host entire app in Elastic Beanstalk. Create custom AMI using Systems Manager Automation which includes all of the required dependencies and web components. Configure the EB env to have an ASG of EC2 instances using multiple AZs w/a LB in front that balances the incoming traffic. Enable **DDB Auto Scaling** and point the app DNS record to the EB load balancer using Route 53.
39 – Launch an EC2 instance w/ the latest Amazon Linux OS in AWS. Use the AWS **VM Import/Export** service to import the EC2 image, export it to a VMware ISO in an S3 bucket and then import the ISO to an on prem server. Once done, commence the testing activity to verify the app's functionalities.
The VM Import/Export enables you to easily import vm images from your existing env to EC2 instances and export them back to your on prem env.
40 – Elastic Beanstalk deployment types
41 – Canary testing EC2 instances w/ASG and ALB:
Prepare another stack that consists of an ALB and ASG which contains the new app version for blue/green environments. Create weighted Alias A records in Route 53 for the two ALBs to adjust the traffic.
Remember – **CodeDeployDefault.LambdaCanary10Percent30Minutes** is specific to Lambdas
42 – Automatically terminate any instance manually logged into via SSH (already have CW Logs agent installed):
Set up CW Logs subscription w/Lambda function which is configured to add a FOR_DELETION tag to the EC2 instance that produced the SSH login event. Run another Lambda every day using the CW Events rule to terminate all EC2 instances w/custom tag for deletion.
NOTE: CW Logs subscription can NOT be directly integrated w/Step Functions



43 – Elastic Beanstalk deployment types
44 – Systems Manager **Automation** offers one-click automations for simplifying complex tasks such as creating golden AMIs (use AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi documents) and recovering unreachable EC2 instances w/ **AWSSupport-ExecuteEC2Rescue** document to recover impaired instances.
Add a tag to each EC2 instance based on their env. Use Systems Manager Automation to preconfigure the AMI by installing all of the required apps and software dependencies. Write a bootstrap script in the User Data to analyze the stage and set the env configuration accordingly. Use the Systems Manager Parameter Store to store the creds as Secure String params.
45 – Capture all rejected or failed approval actions across all pipelines in CodePipeline and send a notification

You can configure notifications to be sent when the state changes for:

- Specified pipelines or all your pipelines. You control this by using `"detail-type": "CodePipeline Pipeline Execution State Change"`.
- Specified stages or all your stages, within a specified pipeline or all your pipelines. You control this by using `"detail-type": "CodePipeline Stage Execution State Change"`.
- Specified actions or all actions, within a specified stage or all stages, within a specified pipeline or all your pipelines. You control this by using `"detail-type": "CodePipeline Action Execution State Change"`.



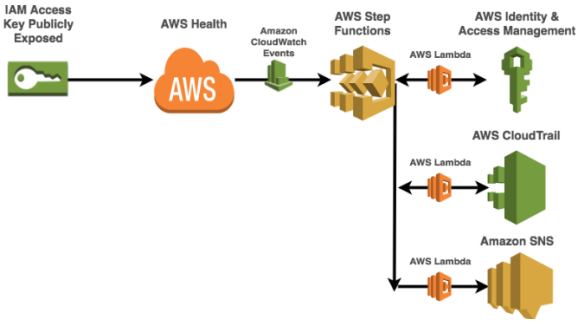
46 – HTTP 403: Access Denied fetching objects

1. S3 bucket policy is not properly configured or IAM user permissions
2. VPC endpoint policy is not properly configured

47 – Pipeline deploying multiple Lambdas in sequence can be refactored by executing actions for each Lambda in **parallel** by setting a configuration that specifies the same **runOrder** value in CodePipeline

48 – AWS proactively monitors popular code repo sites for exposed IAM access keys. On detection of an exposed IAM key, AWS Health generates an **AWS_RISK_CREDENTIALS_EXPOSED** CW Event. In response, you can set up an automated workflow to delete the exposed IAM key, summarize recent API activity for the exposed key and send the summary message to an SNS topic. The CW Event rule event source would be **aws.health**.

You can automate actions in response to new scheduled events for your EC2 instances. For example, you can create CW Events rules for EC2 scheduled events generated by AWS Health.



49 –

Target tracking scaling—Increase or decrease the current capacity of the group based on a target value for a specific metric. This is similar to the way that your thermostat maintains the temperature of your home—you select a temperature and the thermostat does the rest.

Step scaling—Increase or decrease the current capacity of the group based on a set of scaling adjustments, known as *step adjustments*, that vary based on the size of the alarm breach.

Simple scaling—Increase or decrease the current capacity of the group based on a single scaling adjustment.

Scheduled scaling helps you to set up your own scaling schedule according to predictable load changes; sets the desired, minimum, and maximum sizes

50 – Use **OpsWorks** Stacks, create a new stack w/ a custom cookbook and upload the custom recipes to S3. Modify the configuration of the custom recipes to point to S3. Add a new application layer for a standard Node.js application server. Configure the custom recipe to deploy the application using the S3 bucket. Set up time-based instances and attach an IAM role that provides permission to access DynamoDB.

Automatic scaling is based on two instance types, which adjust a layer's online instances based on different criteria:

Time-based instances - They allow a stack to handle loads that follow a predictable pattern by including instances that run only at certain times or on certain days. For example, you could start some instances after 6PM to perform nightly backup tasks or stop some instances on weekends when traffic is lower.

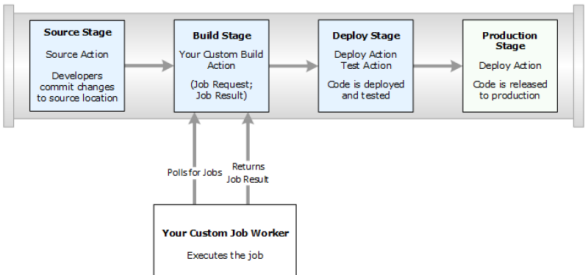
Load-based instances - They allow a stack to handle variable loads by starting additional instances when traffic is high and stopping instances when traffic is low, based on any of several load metrics. For example, you can have AWS OpsWorks Stacks start instances when the average CPU utilization exceeds 80% and stop instances when the average CPU load falls below 60%.

Both time-based and load-based instances are supported for Linux stacks, while only time-based instances are supported for Windows stacks.

51 – Create a pipeline in CodePipeline – set up **custom action** type and create an associated job worker that runs on prem – set the pipeline to invoke the custom action after the source stage – configure the job worker to poll CodePipeline for job requests for the custom action then execute the source code analysis tool and return the status result to CodePipeline

You can create custom actions for the following AWS CodePipeline action categories:

- A custom build action that builds or transforms the items
- A custom deploy action that deploys items to one or more servers, websites, or repositories
- A custom test action that configures and runs automated tests
- A custom invoke action that runs functions



52 – In API Gateway you can create a canary release when deploying API w/canary settings.

You can create one or more aliases for your Lambda. A Lambda alias is like a pointer to a specific Lambda version. Each alias has a unique ARN. An alias can only point to a function version, not another alias. Use routing configuration on an alias to send a portion of traffic to a second function version.

53 – Download all of the external app dependencies from the public internet and store them to an S3 bucket. Set up a **VPC endpoint** for the S3 bucket and then assign an IAM instance profile to the instances in order to allow them to fetch the required dependencies from the bucket.

54 – S3 allows both HTTP and HTTPS requests. To comply with the **s3-bucket-ssl-requests-only** rule, confirm that your bucket policies explicitly deny access to HTTP requests. To determine HTTP or HTTPS requests in a bucket policy, use a condition that checks for the key **"aws:SecureTransport"**. When this key is **true**, this means that the request is sent through HTTPS. To be sure to comply with the **s3-bucket-ssl-requests-only** rule, create a bucket policy that explicitly denies access when the request meets the condition **"aws:SecureTransport": "false"**. This policy explicitly denies access to HTTP requests.

You can enable cross-region replication w/S3.

Encryption: SSE-S3, SSE-KMS, CMK, Client side

55 – In the Audit account, create a new stream in KDS and a Lambda that acts as an event handler to send all of the logs to ElasticSearch cluster. Create a CW subscription filter and use KDS to stream all of the VPC Flow Logs and CW Logs from the sub-accounts to the KDS in the Audit account.

56 - Delay the termination of unhealthy EC2 instances by adding a lifecycle hook to your ASG to move instances in the Terminating state to the Terminating:Wait state. Set up a CW Events rule for the EC2 Instance-terminate Lifecycle Action Auto Scaling Event with an associated Systems Manager Automation document. Trigger the CW agent to push the application logs and then resume the instance termination once all the logs are sent to CW Logs.

57 - Create a single repository in CodeCommit and create a development branch to hold merged changes. Set up CodeBuild to build and test the code stored in the development branch which is triggered to run on every new commit. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using CodeDeploy.

58 - Record configuration changes for EC2 Instances and Dedicated Hosts by turning on the **Config Recording** option in **Config**. Set up a custom Config rule that triggers a Lambda function by using the **config-rule-change-triggered** blueprint. Customize the predefined evaluation logic to verify host placement to return a NON_COMPLIANT result whenever the EC2 instance is not running on a Dedicated Host. Use the Config report to address noncompliant EC2 instances.

59 - Set up two pipelines using CodePipeline for TEST and PROD environments. Configure the PROD pipeline to have a manual approval step. Create a CC repository with a branch for each environment and configure the pipeline to retrieve the source code from CC according to its branch. Deploy the Lambda functions with CF by setting up a deployment step in the pipeline.

60 – DAX caches repeated read requests. CloudFront caches images stored in S3.
61 – A ***projection*** is the set of attributes that is copied from a table into a secondary index. The partition key and sort key of the table are always projected into the index

Features	Global Secondary Index(GSI)	Local Secondary Index(LSI)
Primary Key Schema	Simple (partition key) or composite	Must be composite(partition key and sort key)
Primary Key Attributes	Partition key and sort key(optional) can be any base table attributes of type string, number or binary	Partition key must be the same as base table's partition key. Sort key can be any base table attribute of type string, number, or binary
Size Restrictions	No	For each partition key value, maximum size is 10 GB
Creation	Anytime	When DDB table is created
Deletion	Anytime	When DDB table is deleted
Read Consistency	Eventual consistency	Eventual and strong consistency
Provisioned Throughput Consumption	Each index has its own provisioned throughput independent of base table	Queries, Scans and Updates consume read and write capacity units of the base table
Projected Attributes	Limited to attributes specified during creation	Can request attributes that aren't specified during creation as DDB will fetch them automatically with an extra throughput cost
Count Per Table	20 per DDB table	5 per DDB table

62 - **Trusted Advisor** draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment, and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps.
Integrate CW Events rule and TA to detect the EC2 instances with low utilization. Create a trigger with a Lambda that filters out the reported data based on tags for each env, department, and business unit. Create a second trigger that will invoke another Lambda to terminate the underutilized EC2 instances.
63 - An Aurora global db consists of one primary Region where your data is mastered, and one read-only, secondary Region. Aurora replicates data to the secondary Region. The Aurora cluster in the primary Region where your data is mastered performs both read and write operations. The cluster in the secondary region enables low-latency reads. You can scale up the secondary cluster independently by adding one or more DB instances (Aurora Replicas) to serve read-only workloads. For disaster recovery, you can remove and promote the secondary cluster to allow full read and write operations.
64 - Set up an Config rule with a configuration change trigger that will detect any changes in the S3 bucket configuration and which will also invoke an Systems Manager Automation document with a Lambda that will revert any changes.
65 - Using Systems Manager, create an Automation document with values that configure how the machine image should be created. Launch a new pipeline with a custom action in CodePipeline and integrate it with CW Events to execute the automation document. Build the AMI when the process is triggered. Store the AMI IDs in Systems Manager Parameter Store.
66 - You can add actions to your pipeline that are in a Region different from your pipeline. Build and deployment actions are applicable but NOT source, 3rd party and custom actions.
Create a new pipeline using CodePipeline and a CC repository as the source in your primary region. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the Management Console, set up the pipeline to use cross-region actions that will run the build/deployment actions to other regions. The pipeline will automatically store output files on a default artifact bucket on each region.
67 – Trusted Advisor (NOT Config) can detect under-utilized resources ie; ELB.
- Create a Lambda and integrate it with CW Events. Configure the function to run on a regular basis and to check Trusted Advisor via API. Based on the results, publish a message to an SNS Topic to notify the subscribers.

- Utilize CW Events to monitor Trusted Advisor recommendation results. Set up a trigger to send an email using SNS to notify you about the results of the check.
- Use CW to create alarms on Trusted Advisor metrics in order to detect the load balancers with low utilization. Specify an SNS topic for notification.

68 - Configure and install Systems Manager agent on all of the EC2 instances in your VPC as well as your physical servers on-premises. Use the Systems Manager Patch Manager service and specify the required Systems Manager Resource Groups for your hybrid architecture. Utilize a preconfigured patch baseline and then run scheduled patch updates during maintenance windows.

69 - Set up a new pipeline in CodePipeline and configure a new source code step that will automatically trigger whenever a new code is pushed in CodeCommit. Use CodeBuild for the build step to run the tests automatically then set up an CodeDeploy configuration to deploy the updated Lambda. Select the predefined `CodeDeployDefault.LambdaLinear10PercentEvery3Minutes` configuration option for deployment.

70 – Elastic Beanstalk deployment types

71 – Layer the application server nodes of the cluster using OpsWorks Stacks and add a Chef recipe associated with the `Configure` Lifecycle Event which populates the tdojo-nodes.config file. Set up a configuration which runs each layer's `Configure` recipes that updates the configuration file when a cluster change is detected."

In OpsWorks Stacks Lifecycle Events, each layer has a set of five lifecycle events, each of which has an associated set of recipes that are specific to the layer. When an event occurs on a layer's instance, OpsWorks Stacks automatically runs the appropriate set of recipes. To provide a custom response to these events, implement custom recipes and assign them to the appropriate events for each layer. OpsWorks Stacks runs those recipes after the event's built-in recipes.

When OpsWorks Stacks runs a command on an instance—for example, a deploy command in response to a Deploy lifecycle event—it adds a set of attributes to the instance's node object that describes the stack's current configuration. For Deploy events and Execute Recipes stack commands, OpsWorks Stacks installs deploy attributes, which provide some additional deployment information.

There are five lifecycle events namely: Setup, Configure, Deploy, UnDeploy and Shutdown.

CodeCommit is not a suitable service to use to store your dynamic configuration files

72 - **Application Discovery Service** helps you plan your migration to the AWS cloud by collecting usage and configuration data about your on-premises servers. Application Discovery Service is integrated with AWS Migration Hub, which simplifies your migration tracking. After performing discovery, you can view the discovered servers, group them into applications, and then track the migration status of each application from the Migration Hub console. The discovered data can be exported for analysis in Microsoft Excel or AWS analysis tools such as Athena and QuickSight.

Application Discovery Service offers 2 ways of performing discovery and collecting data about on-premises servers:

1. **Agentless discovery** can be performed by deploying the AWS Agentless Discovery Connector (OVA file) through your VMware vCenter. After the Discovery Connector is configured, it identifies virtual machines (VMs) and hosts associated with vCenter. The Discovery Connector collects the following static configuration data: Server hostnames, IP addresses, MAC addresses, disk resource allocations. Additionally, it collects the utilization data for each VM and computes average and peak utilization for metrics such as CPU, RAM, and Disk I/O. You can export a summary of the system performance information for all the VMs associated with a given VM host and perform a cost analysis of running them in AWS.
2. **Agent-based discovery** can be performed by deploying the AWS Application Discovery Agent on each of your VMs and physical servers. The agent installer is available for both Windows and Linux operating systems. It collects static configuration data, detailed time-series system-performance information, inbound and outbound network connections, and processes that are running. You can export this data to perform a detailed cost analysis and to identify network connections between servers for grouping servers as applications. The Agentless discovery uses the AWS Discovery Connector which is a VMware appliance that can collect information only about VMware virtual machines (VMs). This mode doesn't require you to install a connector on each host. You install the Discovery Connector as a VM in your VMware vCenter Server environment using an Open Virtualization Archive (OVA) file. Because the

Discovery Connector relies on VMware metadata to gather server information regardless of operating system, it minimizes the time required for initial on-premises infrastructure assessment.

73 - In CloudTrail, enable the log file integrity feature on the trail that will automatically generate a digest file for every log file that CT delivers. Verify the integrity of the delivered CT files using the generated digest files.

74 - use Step Functions to launch an EC2 instance for each operating system from the golden AMI, install the **Inspector agent**, and add a custom tag for tracking. Configure the Step Functions to trigger the Inspector assessment for all instances with the custom tag you added right after the EC2 instances have booted up. Trigger the Step Functions every day using an CW Events rule.

75 - use cross-stack references to export resources from a stack so that other stacks can use them. Stacks can use the exported resources by calling them using the Fn::ImportValue function.

Bonso # 2

1 – CloudTrail has data (Get/Delete/PutObject) and management events. After creating a trail you can create a Lambda to log data events for your S3 bucket. Trigger this Lambda using CW Events rule for every action taken on your S3 objects – view logs in CW Logs.

2 – You can use the **AutoScalingRollingUpdate** policy to control how AWS handles rolling updates for an ASG. Using a rolling update has a risk of system outages and performance degredation due to decreased availability of your running EC2 instances. If you want to ensure high availability then use **AutoScalingReplacingUpdate** policy. To find out what's preventing your ASG from updating correctly during a stack update, work through the following scenarios:

- Configure **WaitOnResourceSignals** and PauseTime to avoid problems w/success signals
- Configure **MinSuccessfulInstancesPercent** to avoid stack rollback
- Configure **SuspendProcess** to avoid unexpected changes to the ASG

3 – You can use CW Events to detect and react to changes in the status of Personal Health Dashboard events. Depending on the type of event, you can send notifications via SNS

4 – Create an IAM group for team members and another group for the team leader, both w/**AWSCodeCommitPowerUser** policy attached. Attach another IAM policy to the team members' group that denies Push, Delete and Merge of APIs of CC on the master branch.

5 – Using Systems Manager Parameter Store, upload and manage the db creds w/ a Secure String data type. Create an IAM role w/ an attached policy that allows access and decryption of the db creds. Attach this role to the instance profile of the CodeDeploy managed instances as well as the on prem instances using **register-on-premises-instance** command. Deploy the app packages to the EC2 instance and on prem servers using CodeDeploy

NOTE: Beanstalk can NOT deploy on prem servers; only applicable for EC2

6 – If your EB has an attached RDS DB then you can't use blue/green bc the database would need to be run independently otherwise there would be data loss.

7 – Add a web access control list in front of API Gateway using WAF to block requests that contain malicious SQL code. Track the changes to your web ACLs such as the creation and deletion of rules including the updates to the WAF rule configuration using Config.

8 – CodeDeploy's AppSpec has a default timeout of 1 hour for scripts. You can customize this. If the deployment failed but the logs show the script finished successfully then it likely means CD has reached timeout while waiting for a long running script to finish.

A return value of 0 means success.

9 – Lambda can update S3 simple webpage and send event logs to S3 – can be invoked w/CW Events rule for scale in/out events

10 – Oracle RAC db can be migrated to a large EBS backed EC2 instance and have SSM agent installed to use Systems Manager Patch Manager to automate patch management process – then set up **EBS Data Lifecycle Manager** service to automate the creation of EBS snapshots from EBS volumes of EC2 instances

Aurora and RDS do not support Oracle RAC.

11 – Want to ensure EC2 instances were launched from pre-approved AMI list w/least impact on development process:
Integrate Lambda and CW Events to schedule a daily process that will search through list of running EC2 instances w/in your VPC. Configure the function to determine if any of these are based on unauthorized AMIs. Publish a message to SNS topic to inform and hen automatically terminate EC2 instance.
Use Config w/Lambda that periodically evaluates whether there are EC2 instances launched based on non-approved Amis. Set up remediation action using Systems Manager Automation that will automatically terminate EC2 instance. Publish a message to SNS topic to inform.
12 – Create validation scripts on Lambda and define the functions on the AppSpec lifecycle hook to validate the app using test traffic (AfterAllowTestTraffic). Have CW Alarms trigger an SNS notification when the threshold for 5xx is reached. Associate CW Alarms to your deployment group to have it trigger a rollback when 5xx is reached on CW.
13 – Collect real-time clickstream data using KDS then build and anaylze the sessions using KDA. The aggregated analytics will trigger the real-time events on Lambda and then send them to KDF which in turn, sends data to an S3 bucket. The clickstream data is ingested to a table by a Glue crawler that will be used by Athena.
14 – You can set up a Config Managed Rule and specify a list of approved AMIs. Modify the rule to check whether the running EC2 instances are using specified AMIs. Configure Config to stream configuration changes and notifications to an SNS topic.
15 – Configure CodePipeline to have a trigger to start off the pipeline when a new code change is committed on a certain code branch. In the pipeline, set up the required stages for security review, unit tests, functional tests and manual approval. Use CW Events to detect changes in pipeline stages. Send an email notification to an email using SNS.
16 – Launch a new trail in CT to capture all API calls to your Organizations including calls from Organizations console. Also, track all code calls to Organizations APIs. Integrate CW Events and SNS to raise events when administrator-specified actions occur in an org and configure it to send a notification.
Monitor the compliance of your Organizations using Config. Launch a new SNS topic or CW Events that will send alerts to you for any changes.
17 – In preferred AWS region set up a Global DynamoDB table and enable DynamoDB Streams. Set up replica tables in other regions to replicate. In each local region, store the individual transactions to a DDB replica table in the same region.
18 – You have several ASGs of EC2s and want to update DDB whenever a new instance is created/terminated – configure CW Events target to your ASG that will trigger a Lambda when a lifecycle action occurs. Configure the function to update DDB table w/instance details.
NOTE: Lambda can NOT be a notification target for lifecycle hook of ASG – only CW Events, SNS or SQS
19 – Linux VM hosted on prem recently imported to EC2 – instance running legacy app is difficult to replicate/backup – required to create daily backups using EBS so:
Create a CW Events rule scheduled to run at midnight – set the target to directly call the EC2 CreateSnapshot API to create a snapshot of the needed EBS volumes
20 – You can use CW Events to run ECS tasks when events occur such as PUT/DELETE operations on S3 bucket. For running ECS tasks you don't need a Lambda, you can just use the tasks directly as a target.
21 – Use a NAT Gateway, route credit card payment requests from the EC2 instances in private subnet to the external payment service. Associate an Elastic IP address to the NAT Gateway. Update the route table associated w/ one or more of your private subnets to point Internet bound traffic to the NAT gateway.
22 – Install SSM Agent to all of your EC2 instances (also on prem and VMs). Use Systems Manager Maintenance Windows to automate the patching schedule of your resources. Use Systems Manager Patch Manager to manage and deploy the patches of your EC2 instances based on patch baselines from your on prem network.
23 – If you want to be notified when S3 buckets have the wrong permissions and automatically change then:

- Utilize CW Events to monitor Trusted Advisor security recommendation results and then set a trigger to send an email using SNS to notify you about results
- Set up custom Config rule that checks public S3 bucket permissions. Then send non-compliance notification to your subscribed SNS topic

- Set up custom Config rule to check public S3 buckets permissions and have it send an event on CW Events about policy violations. Have CW Events trigger Lambda to update S3 permissions

24 – Elastic Beanstalk can avoid downtime w/ blue/green deployment where you deploy the new version to a separate environment and then swap CNAMEs of the two environments. Blue/green requires that your env run independently of your database. If your env has an RDS DB attached then it will not transfer over to your second env resulting in data loss.

25 – EC2Rescue can help diagnose and troubleshoot problems on EC2 Linux and Windows server instances. You can run the tool manually or automatically using Systems Manager Automation with the **AWSSupport-ExecuteEC2Rescue** document.

26 – Manage, record and delay the OS security patches of EC2 instances using a combination of Systems Manager and Config.

27 – ASG of EC2 and a separate Linux EC2 instance for batch processing which needs to update its config file based on list of active IP addresses of instances w/in ASG:
Create CW Events rule for the Launch/Terminate events of ASG – set the target to an SSM Run Command that will update the config file on the target EC2 instance

28 – Install the CW Logs agent on EC2 instances to automatically collect and push data to CW Logs. Analyze the log data w/CW Logs Insights.

29 – For ECR/ECS if the latest image is not being deployed when you run the update-service command then add **'--force-new-deployment'** option to CLI command

30 – Launch a Lambda to read the list of proxy IP addresses from S3 bucket. Configure the function to update the ELB security group to allow HTTPS requests only from the given IP addresses. Use the S3 Event notification to automatically invoke the Lambda when the CSV is updated.

31 – Elastic Beanstalk deployment types

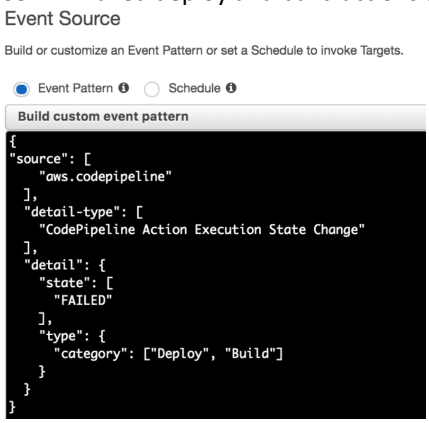
32 – You can use a Lambda to validate part of the CodeDeploy deployment to an ECS app. Define your Lambdas in the AfterAllowTestTraffic hook of the AppSpec. Configure the Lambdas to validate the deployment using the test traffic and rollback if the tests fail.

33 – You can configure a trigger to use SNS to send emails about CodeCommit repo events but they are limited to operational events such as creating branches/pushing code to a branch. You can use a Lambda with CC as a trigger but configure resources w/policy that grants CC permissions to interact w/ resources. Use the Lambda to send event logs to CW Logs.

NOTE: You can't send notifications directly to CW Logs – you must use SNS or a Lambda.

34 - Store the database credentials using Secrets Manager. Encrypt the credentials using KMS. Create an IAM Role for your ECS task execution role and reference it with your task definition which allows access to both KMS and Secrets Manager. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Secrets Manager secret, which contains the sensitive data, to present to the container.

35 – All failed deploy and build actions across all pipelines



You can configure notifications to be sent when the state changes for:

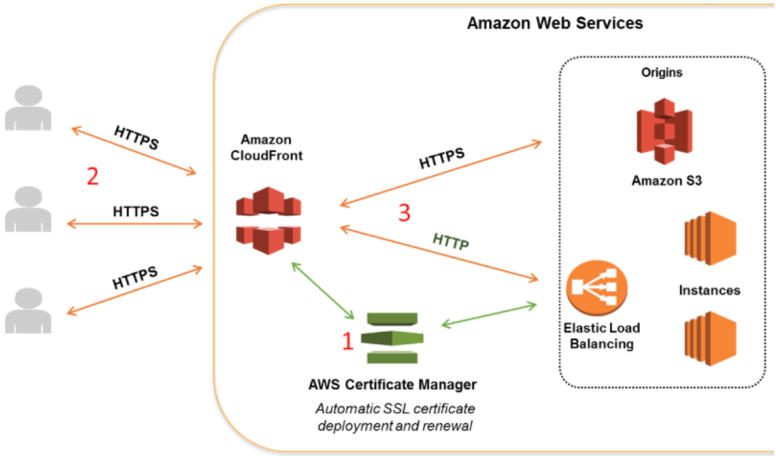
- Specified pipelines or all your pipelines. You control this by using `"detail-type": "CodePipeline Pipeline Execution State Change"`.
- Specified stages or all your stages, within a specified pipeline or all your pipelines. You control this by using `"detail-type": "CodePipeline Stage Execution State Change"`.
- Specified actions or all actions, within a specified stage or all stages, within a specified pipeline or all your pipelines. You control this by using `"detail-type": "CodePipeline Action Execution State Change"`.

36 – AWSCodeCommitFullAccess gives the ability to delete repos; CodeCommitPowerUser can NOT delete repos or create/delete repo related resources in other services such as CW Events.

37 – Automatically detect SSH brute force or malware attacks by enabling GuardDuty in every account. Configure the security account as the GuardDuty Administrator for every member of the organization. Set up a new CW rule in the security account. Configure the rule to send all findings to KDF, which will push the findings to the S3 bucket.
38 – single EC2 dedicated instance w/no ASG due to license restraints w/Aurora – need auto healing to ensure HA if there's outages:
You can configure a CW alarm to automatically recover impaired EC2 instances and notify you through SNS. However, the SNS notification by itself doesn't include the results of the automatic recovery action.
You must also configure an CW Events rule to monitor Personal Health Dashboard (AWS Health) events for your instance. Then, you are notified of the results of automatic recovery actions for an instance.
Set up the Personal Health Dashboard (AWS Health) events to monitor your EC2 Dedicated Instance. Set up a CW Events rule to trigger a Lambda to start a new EC2 instance in an available AZ when the instance status reaches a failure state. Configure an Aurora db with a Read Replica in the other AZ. In the event that the primary database instance fails, promote the read replica to a primary database instance.
39 - Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. You can add tags to your patch baselines themselves when you create or update them.
You can use a *patch group* to associate instances with a specific patch baseline. Patch groups help ensure that you are deploying the appropriate patches, based on the associated patch baseline rules, to the correct set of instances. Patch groups can also help you avoid deploying patches before they have been adequately tested. For example, you can create patch groups for different environments (such as Development, Test, and Production) and register each patch group to an appropriate patch baseline.
When you run AWS-RunPatchBaseline, you can target managed instances using their instance ID or tags. SSM Agent and Patch Manager will then evaluate which patch baseline to use based on the patch group value that you added to the instance.
You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group *must* be defined with the tag key: Patch Group. Note that the key is case-sensitive. You can specify any value, for example, "web servers," but the key must be Patch Group.
The AWS-DefaultPatchBaseline baseline is primarily used to approve all Windows Server operating system patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". Patches are auto-approved seven days after release.
Set up two Patch Groups with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up two non-overlapping maintenance windows and associate each with a different patch group. Register targets with specific maintenance windows using Patch Group tags. Assign the AWS-RunPatchBaseline document as a task within each maintenance window which has a different processing start time is the correct answer as it properly uses two Patch Groups, non-overlapping maintenance windows and the AWS-DefaultPatchBaseline baseline to ensure that the EC2 instance reboots do not occur at the same time.
40 – The s3-bucket-public-read-prohibited and s3-bucket-public-write-prohibited are **Config managed rules** that check if your S3 buckets do not allow public read access and write access. These rules check the Block Public Access settings, the bucket policy, and the bucket access control list (ACL).
Since they are managed rules, we can simply use them on the fly. Without these rules, we would need to implement our own custom-managed rule using a Lambda. The next step would be to configure a CW Events rule that watches for Config events whenever it detects a non-compliant S3 bucket. Finally, we specify an SNS topic as a trigger for CW events to send alerts to the team.
41 - Global secondary index — an index with a partition key and a sort key that can be different from those on the base table.
Local secondary index — an index that has the same partition key as the base table, but a different sort key.
42 – Strongly consistent reads are NOT supported on GSI.
You can NOT add a LSI to an already existing table – migrate the data from the existing table to the new table

43 - You can aggregate your data before you publish it to CW. When you have multiple data points per minute, aggregating data minimizes the number of calls to **put-metric-data**.
Configure the instances to push the number of payments and refund transactions to CW as a custom metric. Set up a CW alarm to notify the DevOps team using SNS when the threshold is breached. View statistical graphs of your published metrics with the Management Console.
44 – Using Elastic Beanstalk, host the multi-tier Node.js web portal in a load-balancing and autoscaling environment. Set up an RDS MySQL database with a Multi-AZ deployments configuration that is decoupled from the EB stack. Set the log options to stream the application logs to CW Logs with a 90-day retention.
45 – Decouple the Amazon RDS instance from your Elastic Beanstalk environment using the blue/green deployment strategy to decouple. Take an RDS DB snapshot of the database and enable deletion protection. Set up a new Elastic Beanstalk environment with the necessary information to connect to the RDS instance. Before terminating the old Elastic Beanstalk environment, remove its security group rule first before proceeding. If the existing security group rule is not removed it will hinder the deletion of the old environment.
46 – Macie is an ML-powered security service that helps you prevent data loss by automatically discovering, classifying, and protecting sensitive data stored in S3. Macie uses machine learning to recognize sensitive data such as personally identifiable information (PII) or intellectual property, assigns a business value, and provides visibility into where this data is stored and how it is being used in your organization.
47 - As part of the deployment process, the CodeDeploy agent removes from each instance all the files installed by the most recent deployment. If files that weren't part of a previous deployment appear in target deployment locations, you can choose what CD does with them during the next deployment:
Fail the deployment — An error is reported and the deployment status is changed to Failed.
Overwrite the content — The version of the file from the application revision replaces the version already on the instance.
Retain the content — The file in the target location is kept and the version in the application revision is not copied to the instance.
48 – Launch a brand new OpsWorks stack that contains a new layer with the latest PCI-DSS compliant web portal version. Use a Blue/Green deployment strategy with Route 53 that shifts traffic between the existing stack and new stack. Route only a small portion of incoming production traffic to use the new application stack while maintaining the old application stack. Check the features of the new portal and once it's 100% validated, slowly increase incoming production traffic to the new stack. Change the Route 53 to revert to old stack if there are issues on the new stack.
49 - Use Shield Advanced to enable enhanced DDoS attack detection and monitoring for application-layer traffic of the company's AWS resources. Ensure that every security group in the VPC only allows certain ports and traffic from authorized servers or services. Protect your origin servers by putting it behind a CloudFront distribution.
Set up WAF rules that identify and block common DDoS request patterns to effectively mitigate a DDoS attack on the company's cloud infrastructure. Ensure that the Network Access Control Lists (ACLs) only allow the required ports and network addresses in the VPC.
50 - Add a test action to the pipeline to run unit and functional tests using CodeBuild. Verify that the test results passed before deploying the new application revision to the PROD env.
In the pipeline, add an CodeDeploy action to deploy the latest version of the application to the PRE-PROD environment. Set up a manual approval action in the pipeline so that the QA team can perform the required tests. Add another CodeDeploy action that deploys the verified code to the PROD environment after the manual approval action.
51 – The **X-Ray SDK** does not send trace data directly to X-Ray. To avoid calling the service every time your application serves a request, the SDK sends the trace data to a daemon, which collects segments for multiple requests and uploads them in batches.
To properly instrument your applications in ECS, you have to create a Docker image that runs the X-Ray daemon, upload it to a Docker image repository, and then deploy it to your ECS cluster. You can use port mappings and network mode settings in your **task definition file** (NOT the container agent) to allow your application to communicate with the daemon container.
The X-Ray daemon is a software application that listens for traffic on **UDP** port **2000**, gathers raw segment data, and relays it to the X-Ray API.

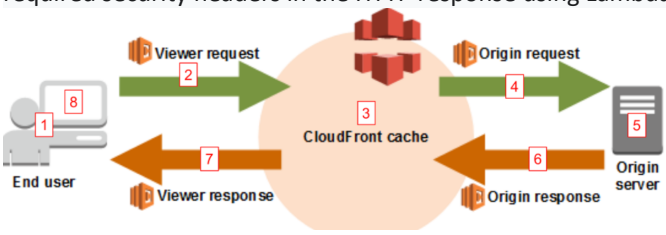
52 – Launch a new ElastiCache for Redis cluster with nodes across multiple Availability Zones. Configure the application to store user-session information in ElastiCache.
53 – Generate an SSL certificate that is signed by a trusted third-party certificate authority in the ALB. Import the certificate into Certificate Manager. Set the Viewer Protocol Policy to **HTTPS Only** in CloudFront and then use an SSL/TLS certificate from a 3rd party certificate authority which was imported to either Certificate Manager or the IAM certificate store.
NOTE: You can not directly upload a self signed cert in your ALB.
NOTE: **Match Viewer** policy communicates w/custom origin using HTTP OR HTTPS



54 – Tag each instance based on its environment, business unit, and operating system. Set up a patch baseline in Systems Manager Patch Manager for each environment. Categorize each EC2 instance based on its tags using Patch Groups. Apply the required patches specified in the corresponding patch baseline to each Patch Group.
55 – To restrict access to content that you serve from S3 buckets, you create CloudFront signed URLs or signed cookies to limit access to files in your S3 bucket, and then you create a special CloudFront user called an origin access identity (OAI) and associate it with your distribution. Then you configure permissions so that CloudFront can use the OAI to access and serve files to your users, but users can't use a direct URL to the S3 bucket to access a file there.
56 – Launch a read replica of the primary database to the second region. Set up **RDS Event Notification** to publish status updates to an SNS topic. Create a Lambda subscribed to the topic to monitor database health. Configure the Lambda to promote the read replica as the primary in the event of a failure. Update the Route 53 record to redirect traffic from the primary region to the secondary region.
57 – **Lambda@Edge** runs your code in response to events generated by the CloudFront CDN. Lambda@Edge lets you run Node.js Lambda functions to customize content that CF delivers, executing the functions in AWS locations closer to the viewer.
58 – In the given scenario, you can use Lambda@Edge to allow your Lambda functions to customize the content that CloudFront delivers and to execute the authentication process in AWS locations closer to the users. In addition, you can set up an origin failover by creating an origin group with two origins with one as the primary origin and the other as the second origin, which CloudFront automatically switches to when the primary origin fails. This will alleviate the occasional HTTP 504 errors that users are experiencing.
59 – you still want to restrict the port 22 access from the Public IP of your on-premises data center only. With this, you want to be notified of any security check recommendations from Trusted Advisor and automatically solve the non-compliance based on the results:

- Create a Lambda and integrate CW Events and Lambda to execute the function on a regular schedule to check Trusted Advisor via API. Based on the results, publish a message to an SNS Topic to notify the subscribers.
- Set up custom Config rule that checks security groups to make sure that port 22 is not open to public. Send a notification to an SNS topic for non-compliance.
- Set up custom Config rule to execute a remediation action using Systems Manager Automation to update the publicly open port 22 on the instances and restrict to only your office's public IP.

60 – By default, an alias points to a single Lambda function version. When the alias is updated to point to a different function version, incoming request traffic in turn instantly points to the updated version. This exposes that alias to any potential instabilities introduced by the new version. To minimize this impact, you can implement the routing-config parameter of the Lambda alias that allows you to point to two different versions of the Lambda function and dictate what percentage of incoming traffic is sent to each version.
With the introduction of **alias traffic shifting**, it is now possible to trivially implement canary deployments of Lambda functions. By updating additional version weights on an alias, invocation traffic is routed to the new function versions based on the weight specified. Detailed CloudWatch metrics for the alias and version can be analyzed during the deployment, or other health checks performed, to ensure that the new version is healthy before proceeding.
https://aws.amazon.com/blogs/compute/implementing-canary-deployments-of-aws-lambda-functions-with-alias-traffic-shifting/
61 – **GuardDuty** offers threat detection that enables you to continuously monitor and protect your AWS accounts and workloads. GD analyzes continuous streams of meta-data generated from your account and network activity found in CT Events, VPC Flow Logs, and DNS Logs. It also uses integrated threat intelligence such as known malicious IP addresses, anomaly detection, and machine learning to identify threats more accurately. This can include issues like escalations of privileges, uses of exposed credentials, or communication with malicious IPs, URLs, or domains.
62 – Centrally manage all of your accounts using Organizations. Group your accounts, which belong to a specific business unit, to individual Organization Units (OU). Set up an IAM Role in the production account which has a policy that allows access to the EC2 instances including resource-level permission to terminate the instances owned by a particular business unit. Associate the cross-account access and the IAM policy to every member accounts of the OU.
NOTE: The AWSServiceRoleForOrganizations service-linked role is primarily used to only allow Organizations to create service-linked roles for other AWS services. This service-linked role is present in all organizations and not just in a specific OU.
63 – Create a patch baseline in Systems Manager Patch Manager that defines which patches are approved for installation on your instances. Set up the Config Managed Rule which automatically checks whether your running EC2 instances are using approved AMIs. Create CW Alarms to notify you if there are any non-compliant instances running in your VPC.
64 – Host the application on an S3 bucket configured for website hosting. Set up a CloudFront web distribution and set the S3 bucket as the origin with the origin response event set to trigger a Lambda@Edge function. Add required security headers in the HTTP response using Lambda.



65 – You can configure alarms for several scenarios in CloudTrail events. In this case, you can create an CW alarm that is triggered when an S3 API call is made to PUT or DELETE bucket policy, bucket lifecycle, bucket replication, or to PUT a bucket ACL. A CT trail is required since it will send its logs to a CW Log group. To create an alarm, you must first create a metric filter and then configure an alarm based on the filter.
66 – You can configure a CodeCommit repository so that code pushes or other events trigger actions, such as sending a notification from SNS or invoking a function in Lambda.
You must create and then configure resources with a policy that grants CC the permissions to interact with those resources. You must create the resource in the same Region as the CC repo.
67 – If your updated Docker image uses the same tag as what is in the existing task definition for your service (for example, `my_image:latest`), you do not need to create a new revision of your task definition. You can update your service with your custom configuration, keep the current settings for your service, and select **Force new deployment**. The new tasks launched by the deployment pull the current image/tag combination from your

repository when they start. The **Force new deployment** option is also used when updating a Fargate task to use a more current platform version when you specify LATEST.

68 –**Storage Gateway** offers file-based, volume-based, and tape-based storage solutions. With a tape gateway, you can cost-effectively and durably archive backup data in GLACIER or DEEP_ARCHIVE. A tape gateway provides a virtual tape infrastructure that scales seamlessly with your business needs and eliminates the operational burden of provisioning, scaling, and maintaining a physical tape infrastructure.

Connect the on-premises file system to Storage Gateway by setting up a file gateway appliance on-premises. Use the MAM solution to extract the media files from the current data store and send them into the file gateway. Populate a collection using Rekognition by building a catalog of faces from the processed media files. Launch a Lambda to invoke Rekognition Javascript SDK to have it fetch the media files from the S3 bucket which is backing the file gateway. Retrieve the needed metadata using the Lambda and store the information into the MAM solution.

69 – Elastic Beanstalk deployment types

70 – To increase your cache hit ratio, you can configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age. The shorter the cache duration, the more frequently CloudFront forwards another request to your origin to determine whether the object has changed and, if so, to get the latest version.

71 – Use Systems Manager Parameter Store in conjunction with CloudFormation to retrieve the latest AMI IDs for your template. Call the update-stack API in CloudFormation in your template whenever you decide to update the EC2 instances.

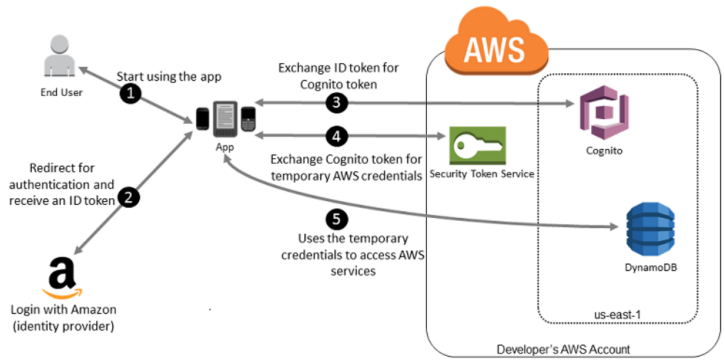
72 – **EC2 Inspector** is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices. After performing an assessment, Inspector produces a detailed list of security findings prioritized by level of severity. These findings can be reviewed directly or as part of detailed assessment reports which are available via the EC2 Inspector console or API. Inspector security assessments help you check for unintended network accessibility of your EC2 instances and for vulnerabilities on those EC2 instances.

Use EC2 Inspector for automated security assessment to help improve the security and compliance of your applications. Launch an ASG of EC2 instances on three AZs. Set up an ALB to distribute the incoming traffic. Set up an Aurora Multi-Master as the database tier. Using Route 53, create an alias record for the root domain to point to the load balancer.

73 – Instruct the Security team to set up an CloudFormation stack that creates an CodePipeline pipeline that builds new AMIs. Then, store the AMI ARNs as parameters in Systems Manager Parameter Store as part of the pipeline output. Order the DevOps team to use the AWS::SSM::Parameter section in their CloudFormation stack to obtain the most recent AMI ARN from the Parameter Store.

74 – Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with social identity providers like Facebook, Google or any other OpenID Connect (OIDC)-compatible IdP. Create a new IAM Role and grant permissions to allow access to S3 and DynamoDB. Configure the mobile application to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table.

Create an identity pool in Cognito that will be used to store the end-user identities organized for your mobile app. Cognito will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for Cognito users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use Cognito. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client.



75 - Monitor the CodeDeploy notifications when a code deployment has been successfully completed using CW Events. Configure CW Events to trigger an EC2 Inspector scan to your instances.

Marrek

1 – Aurora: self healing, fault tolerant, auto scaling, NOT in-memory; GLOBAL – replicates data for fast local reads w/Disaster Recovery

2 – workflow to detect AWS account creds on public Github, report of API calls – CW Event checking AWS_RISK_CREDENTIALS_EXPOSED in Health service – trigger Step Function to issue API calls to IAM, CT and SNS to achieve desired reqs (DeleteAccessKeyPair, LookupCTEvents, NotifySecurity)

3 – S3 encryption at rest and in transit – bucket policy condition to deny request '**aws:SecureTransport: false'** and encrypt w/**SSE-S3 (KMS may result in throttling)**; S3 Cross Region Replication

4 – caching S3 – Cloudfront and DDB – DAX

5 – use Config to track resources, tag resources, use SNS to stream changes to Lambda that writes to S3, create QuickSight dashboard on top of it

6 – If you wanted to use EC2 instances to facilitate a deployment – create EC2 instance w/IAM role giving access to S3 bucket where CodeDeploy is deploying from, ensure EC2 instance also has CodeDeploy agent installed, tag instance to have it part of deployment group

7 – buildspec.yml looking for env var **CODEBUILD_SOURCE_VERSION** at runtime, use variable in **artifacts** section; to use CodeBuild to be named based on branch being tested and scalable to newer branches (so don't use a template for EACH branch)

8 – **Config** managed rule checking **EBS encryption**, CW Event rule to provide alerting (w/SNS as target – note SNS topics in Config can only be used to stream all notifications and config changes; for a single rule you have to use Events)

9 – CloudFormation update ASG instances use **AutoScalingRollingUpdate**

10 – Manage EC2/on prem instances using OpsWorks – include chef cookbook on the configure lifecycle event that will update the config file

A stack is top level OpsWorks entity representing a set of instances to manage; each stack contains layer(s) – each representing a stack component

https://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-events.html

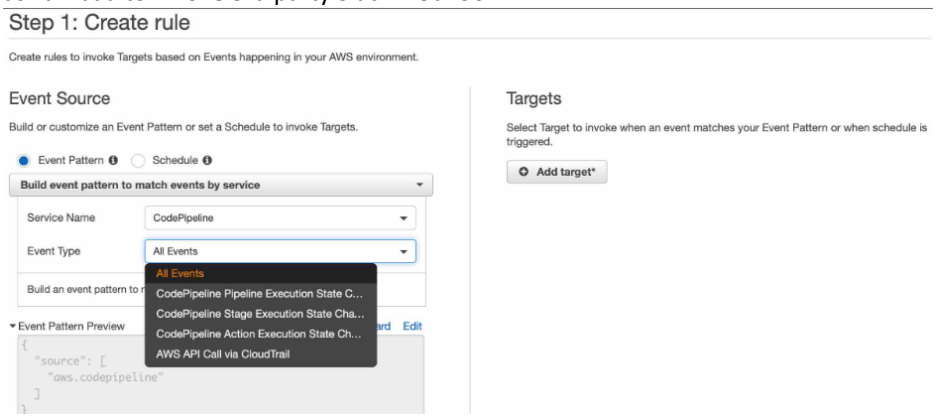
Stack Lifecycle Events: Setup, Configure, Deploy, Undeploy, Shutdown

11 – In a master account create an AMI and share it across other regions – Systems Manager Automation document.

Config Custom Rule in all accounts w/CF StackSets to track/audit compliance of AMI usage across accounts – report the rule's results using Config aggregation.

12 – ECS tasks can include the **awslogs** driver and write to CW Logs natively but the EC2 instance will be the one to write to CW so EC2 must have an Instance Role w/permissions

13 – ASG keeps terminating instances right after boot from health check – to troubleshoot one instance set it to **Standby** right after launched – creating an autoscaling hook for instance termination would work BUT comes with one hour default timeout
14 – CF does not detect a new file has been uploaded to S3 unless one of these parameters changes – S3Bucket, S3Key, S3ObjectVersion
15 – API Gateway lets you use mapping templates to map payload from method request to corresponding integration request and from an integration response to corresponding method response – to add a static, missing field to the mapping template you can create different API versions (stages) – one for the old and one for new traffic – both can use the same Lambda function; mapping templates do not support adding default values for fields as these only support static fields
16 – To send all CodePipeline failures to be sent to Slack channel – CW Event Rule with source as 'aws.codepipeline', detail-type as 'CodePipeline Execution State Change' and detail:state as 'FAILED' – with target as Lambda to invoke 3rd party Slack webhook



17 – Add a new IAM policy attached to the group to Deny codecommit:GitPush with a condition on master branch
18 – Upgrade MySQL db managed by CF and setup Multi-AZ – create RDS read replica in CF template by specifying **SourceDBInstanceIdentifier**, after creation upgrade replica **EngineVersion**, then promote replica as new master
19 – CodeDeploy use appspec.yml BeforeAllowTraffic to check completion of Step Function (for Lambda deployment)
20 – CodeDeploy deploying EC2 instances w/ALB – gradual deployment and need to auto rollback if high CPU utilization for EC2 – create CW Metric and Alarm of it integrated w/CodeDeploy and enable rollback; CodeDeploy can also rollback when a deployment fails; CodeDeploy only rolls back w/Alarms, not metrics
21 – CloudFormation template w/Lambda, S3, IAM role and DDB getting **InsufficientCapabilitiesException** – enable IAM Capability on CodePipeline config for Deploy CF stage action bc CF stack is trying to create IAM role but doesn't have those specified capabilities
You can use IAM w/CF to control what users can do w/CF such as view stack templates, create/delete stacks
22 – SSM Patch Manager automates patching instances and (default or custom) baselines for greater control over which patches are approved/rejected
23 – All objects must be deleted from S3 bucket or remove all instances in EC2 security group before deleting
24 – CodeDeploy rolling update w/no downtime to EC2 instances running behind ALB, rollback in case of failed health checks – **ValidationService** hook verification – the last deployment lifecycle event to verify deployment was successful, configure CD to rollback on fail

List of Lifecycle Event hooks for EC2 deployment:

List of lifecycle event hooks

An EC2/On-Premises deployment hook is executed once per deployment to an instance. You can specify one or more scripts to run in a hook. Each hook for a lifecycle event is specified with a string on a separate line. Here are descriptions of the hooks available for use in your AppSpec file.

For information about which lifecycle event hooks are valid for which deployment and rollback types, see Lifecycle event hook availability.

- `ApplicationStop` – This deployment lifecycle event occurs even before the application revision is downloaded. You can specify scripts for this event to gracefully stop the application or remove currently installed packages in preparation for a deployment. The AppSpec file and scripts used for this deployment lifecycle event are from the previous successfully deployed application revision.

  Note
  An AppSpec file does not exist on an instance before you deploy to it. For this reason, the `ApplicationStop` hook does not run the first time you deploy to the instance. You can use the `ApplicationStop` hook the second time you deploy to an instance.

  To determine the location of the last successfully deployed application revision, the CodeDeploy agent looks up the location listed in the *deployment-group-id*`_last_successful_install` file. This file is located in:

  `/opt/codedeploy-agent/deployment-root/deployment-instructions` folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

  `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` folder on Windows Server Amazon EC2 instances.

  To troubleshoot a deployment that fails during the `ApplicationStop` deployment lifecycle event, see Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events.
- `DownloadBundle` – During this deployment lifecycle event, the CodeDeploy agent copies the application revision files to a temporary location:

  `/opt/codedeploy-agent/deployment-root/`*deployment-group-id*`/`*deployment-id*`/deployment-archive` folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

  `C:\ProgramData\Amazon\CodeDeploy\`*deployment-group-id*`\`*deployment-id*`\deployment-archive` folder on Windows Server Amazon EC2 instances.

  This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

  To troubleshoot a deployment that fails during the `DownloadBundle` deployment lifecycle event, see Troubleshooting a failed DownloadBundle deployment lifecycle event with UnknownError: not opened for reading.
- `BeforeInstall` – You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.
- `Install` – During this deployment lifecycle event, the CodeDeploy agent copies the revision files from the temporary location to the final destination folder. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.
- `AfterInstall` – You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.
- `ApplicationStart` – You typically use this deployment lifecycle event to restart services that were stopped during `ApplicationStop`.
- `ValidateService` – This is the last deployment lifecycle event. It is used to verify the deployment was completed successfully.
- `BeforeBlockTraffic` – You can use this deployment lifecycle event to run tasks on instances before they are deregistered from a load balancer.

  To troubleshoot a deployment that fails during the `BeforeBlockTraffic` deployment lifecycle event, see Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events.
- `BlockTraffic` – During this deployment lifecycle event, internet traffic is blocked from accessing instances that are currently serving traffic. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.
- `AfterBlockTraffic` – You can use this deployment lifecycle event to run tasks on instances after they are deregistered from a load balancer.

  To troubleshoot a deployment that fails during the `AfterBlockTraffic` deployment lifecycle event, see Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events.
- `BeforeAllowTraffic` – You can use this deployment lifecycle event to run tasks on instances before they are registered with a load balancer.
- `AllowTraffic` – During this deployment lifecycle event, internet traffic is allowed to access instances after a deployment. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.
- `AfterAllowTraffic` – You can use this deployment lifecycle event to run tasks on instances after they are registered with a load balancer.

via - https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html

Lifecycle Event hooks availability for EC2 deployment and rollback scenarios:

Lifecycle event hook availability

The following table lists the lifecycle event hooks available for each deployment and rollback scenario.

Lifecycle event name	In-place deployment[1]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓		✓		
DownloadBundle[2]	✓		✓		
BeforeInstall	✓		✓		
Install[2]	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic[2]	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic[2]	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	

[1] Also applies to the rollback of an in-place deployment.

[2] Reserved for CodeDeploy operations. Cannot be used to run scripts.

via - https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-

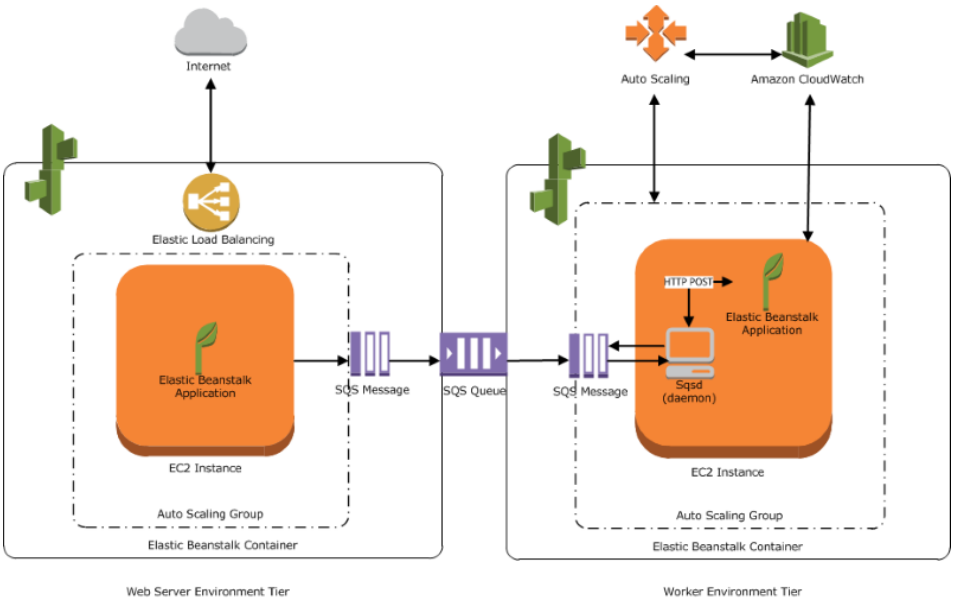
25 – Use CodePipeline **runOrder** for parallel actions (ie: CB test scripts) – default value is 1

26 – AMIs can be created w/SSM Automation documents – the AMI ID can be stored in SSM Parameter Store in one region and have a lambda copy it across other regions

27 – Track API calls via CloudTrail using CW Event rules and use SNS as target

28 – You can use Elastic Beanstalk config files (.ebextensions) w/your web app's source code to configure env and customize AWS resources it contains – you may want to customize and configure the software your app depends on using the **commands** key to execute commands on the EC2 instance – the commands run before the app/web server are set up and the app version file is extracted – you can use **container_commands** key to execute commands that affect your app source code – container commands run after the app/web server have been set up and the app version archive has been extracted but before the app version is deployed – you can use **leader_only** to only run the command on a single instance or configure a test to only run the command when a test command evaluates to **true** – if you specify a commands block then every EC2 instance will run it and does not support the leader_only attribute

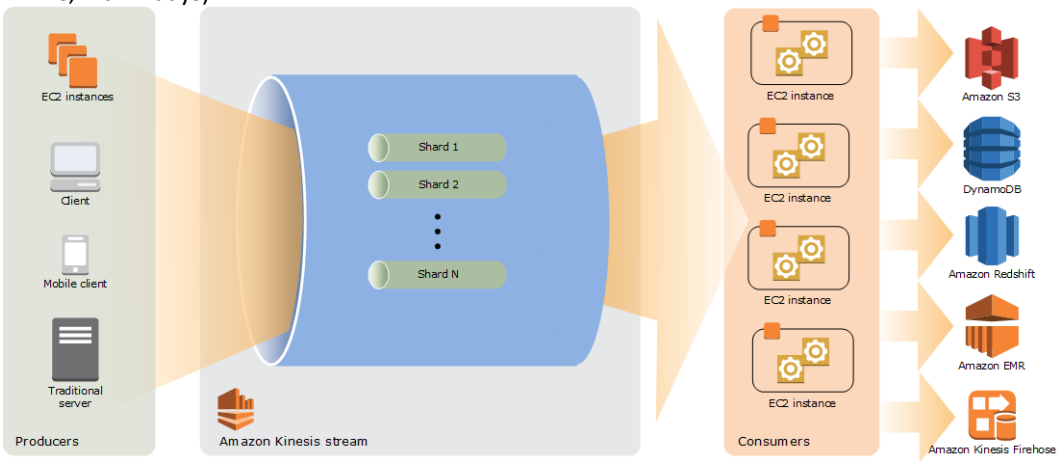
29 – Store Golden AMI id in SSM Parameter Store, create CF parameter that points to the SSM Param Store and is passed on to the config of the EB env, create a CW Event rule triggered every week that launches a Lambda that triggers a refresh of all CF templates using UpdateStack API; CF cannot source params from S3
30 – Determine whether a log file was modified, deleted or unchanged after CloudTrail delivered it using **log file integrity validation** – when enabled a hash is created for every log filed deliverd as a digest file – use **validation-logs** command
31 – To perform blue/green in Elastic Beanstalk deploy to a new env and do a CNAME swap, the swap is not supported by CF so you need to create a custom Lambda to perform the API call to invoke as part of a Custom job in CodePipeline
32 – You can use metric streams to continually stream CW metrics to S3 using Kinesis Firehose, create QuickSight dashboard to visualize the metrics and use Athena to query specific time ranges
KDF delivery stream must trust CW through an IAM role that has permissions to KDF
33 – Create CW Event Rule that reacts to the creation and updates done to PRs in source repo, the target of that rule should be CB, create a second CW Event Rule to watch CB build success/failure event and as target invoke a Lambda to update the PR w/build outcome
34 – Macie protects S3 sensitive data and can use CW Events w/SNS
35 – set CICD pipeline to deploy master, use feature branches w/PRs, set IAM policy on dev group to prevent direct pushes to master
36 – Create CF template w/ an ASG of min/max capacity of 1 and an EBS volume, tag the ASG and EBS volume, create a User Data script that will acquire the EBS volume at boot time, use a master CF template and reference the nested template 6x
37 – copy artifacts from one pipeline to another w/S3 which can be included as a deploy action
38 – ASG lifecycle hook for terminate actions and CW Event rule for that hook to invoke Lambda which uses SSM Run Command to extract logs and store in S3; CW Logs agent only used for continuous streaming and not one time log extract
39 – for languages not supported by EB use single/multi container support w/ECR using a dockerrun.aws.json file
40 – ASG scaling policy can track CPU utilization as target; scheduled action at specified times
41 – CW Logs Metric Filter w/ CW Alarm linked to metric (ie: error patterns) – SNS as target
42 – ECS SHA Tracking provides visibility and identification to track where container images are deployed by using task state change events emitted to CW Events – so if ECS instances do not detect newer image version then specify sha256 so newer versions have different has value
43 – can create CW Event (custom or scheduled)rules on CC repo reacting to pushes – target can be Lambda
44 – enable Config in all accounts/regions w/Config aggregator in central account – CT compliance breaches would generate CW Events subscribed by Lambda to send out notificaitons
45 – Issue bc use of second ELB for the second app stack and then changing the DNS route to direct the traffic to the other stack when required; replace only infrastructure behind ELB – migrate to one ALB and just use one target group at a time behind each ASG for correct routing
46 – CodeBuild stage can acquire ECR credentials using CLI helpers, build the image and push to ECR; CodeDeploy stage with a target being ECS service; CodeDeploy/CF can support blue/green deployments for ECS; CW Events does NOT support CodeDeploy as a target – must be invoked via CodePipeline; ECR credentials must be acquired using IAM roles and CLI helpers on Codebuild
47 – Secrets Manager (does not support DDB creds), IAM instance role for EC2 to access
48 – RPO (Recovery Point Objective) data loss and RTO (Recovery Time Objective) down time; create custom metric via the app that captures the lag in file reads and then uses it for scaling the ASG managing the EC2 instances to replicate the source EFS cluster into S3 – use another ASG to copy data from S3 into EFS in the target region – needing minimum RPO so continuous replication and minimum RTO so hot EFS system ready to go
49 – Beanstalk env that's a **worker** env to process invoices via SQS; invoices uploaded to S3; worker tier processes files and cron job defined using cron.yml will send out emails; worker tier must be sep env from web tier and cron.yml must be defined on worker tier



50 – For cross account log data sharing with subscriptions you can collaborate w/an owner of a different account and receive their log events on your resources such as Kinesis or KDF

51 – **Golden AMI** contains web app – configure dynamic part at runtime using EC2 User Data script

52 – KCL ensures for every shard there is a record processor running/processing for that shard; KCL may need to be run on multiple EC2 instances behind ASG based on metric **MillisBehindLatest**; also increase **the stream data retention period** – this is the length of time data records are accessible after they are added to the stream (default 24 hrs, max 7 days)



53 – Inspector can NOT run an assessment directly on an AMI, it will not launch an EC2 instance for you; so to do a daily assessment on AMIs use a CW Event w/daily schedule – target being a Step Function which will launch an EC2 instance from the AMI and tag it w/CheckVulnerabilities: True – SF starts an AMI assessment template using Inspector – terminate instance afterwards

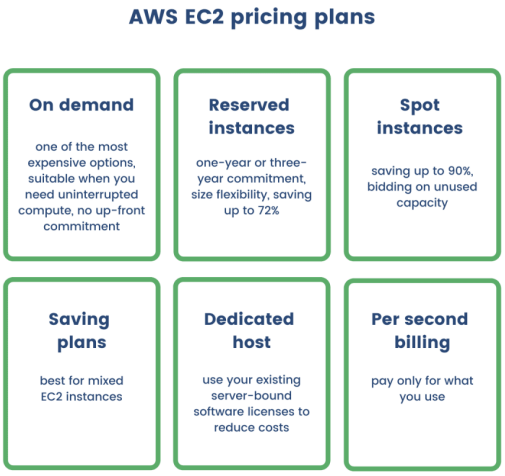
54 – CC repo with 2 pipelines – one deploys changes to master branch to dev and staging envs; the other deploys changes to prod branch to prod env

55 – **SSM Agent** can be installed on EC2, on prem or VM. **SSM Inventory** can be used to cllect metadata from your managed instances – store data in S3

56 – CodePipeline pointing to master branch of CC and auto deploy to staging env using CodeDeploy; After that stage invoke a CodeBuild that runs mandatory test suite; if stage doesn't fail then deploy to PROD

57 – Config managed rule to track if CloudTrail is enabled (cloudtrail-security-trail-enabled); CW Event rule to get alerted in case of change and trigger Lambda to re-enable CT

58 – **API Gateway Stage, Canary deployments** on v1 stage, deploy new stage to v1 stage and assign a small amount of traffic to canary stage, track metrics w/CW; when a new API route is implemented you must create a new API Gateway stage and NOT a **Lambda alias** which are only used to update the behavior of an existing route (one route in API Gateway is mapped to one Lambda function)
59 – EC2 Dedicated Host is physical server fully dedicated for your use, when you bring your own licenses you can let AWS take care of all admin tasks on your behalf, access to CPU sockets
Config – managed and custom rules; custom rules: configuration and periodic types



60 – OpsWorks – CW Event rule for ‘aws.opsworks’ and set initiated_by field to ‘auto-healing’; target a Lambda to send notifications out to Slack; OpsWorks does NOT send notifications directly to SNS
61 – CodePipeline which every push to CC master branch gets deployed to dev, stage, prod EC2 envs – when deploying to prod the traffic should be deployed to few instances so metrics can be gathered before manual approval step – CodeDeploy needs 4 deployment groups: dev, stage, canary testing instances in prod and one for entire prod instances – create ONE CodePipeline and chain up these stages w/manual approval step after canary instances
Separate pipelines won’t allow for a manual approval; CodeDeploy has a Canary Deployment setting but it’s ONLY for Lambda and ECS

The exam may try to trap you on some of the following details on deployment-related processes. Be aware of what's possible.

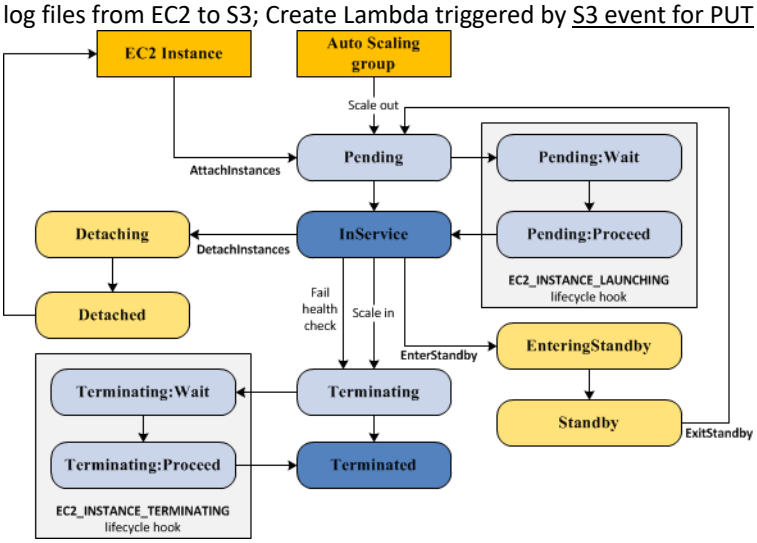
- A deployment group contains individually tagged Amazon EC2 instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.
- Deployments that use the EC2/On-Premises compute platform manage the way in which traffic is directed to instances by using an in-place or blue/green deployment type. During an in-place deployment, CodeDeploy performs a rolling update across Amazon EC2 instances. During a blue/green deployment, the latest application revision is installed on replacement instances.
- If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.
- You CANNOT use canary, linear, or all-at-once configuration for EC2/On-Premises compute platform.
- You can manage the way in which traffic is shifted to the updated Lambda function versions during deployment by choosing a canary, linear, or all-at-once configuration.
- You can deploy an Amazon ECS containerized application as a task set. You can manage the way in which traffic is shifted to the updated task set during deployment by choosing a canary, linear, or all-at-once configuration.
- Amazon ECS blue/green deployments are supported using both CodeDeploy and AWS CloudFormation. For blue/green deployments through AWS CloudFormation, you don't create a CodeDeploy application or deployment group.
- Your deployable content and the AppSpec file are combined into an archive file (also known as application revision) and then upload it to an Amazon S3 bucket or a GitHub repository. Remember these two locations. AWS Lambda revisions can be stored in Amazon S3 buckets. EC2/On-Premises revisions are stored in Amazon S3 buckets or GitHub repositories.
- AWS Lambda and Amazon ECS deployments CANNOT use an in-place deployment type.

62 – ALB health checks returns protocol and response code; if your app depends on a db you can enhance the health check so the return status code corresponds to the connectivity of the db
63 – Jenkins must be deployed as a multi-master across multi-AZ to be highly available and fault tolerant, Jenkins CodeBuild plugin (don't use Jenkins w/EC2 bc it'd be expensive)
64 –**Trusted Advisor** w/**check for low-utilized EC2 instances**, CW Event that tracks events created by Trusted Advisor and use Lambda as target for that event, Lambda triggers an **SSM Automation doc** w/ manual approval step to terminate
65 – If an EC2 Auto Scaling scale up event occurs while a deployment is underway then the new instance will be updated w/ the app revision that was most recently deployed not the app revision currently being deployed – if deployment succeeds the old instances and the newly scaled up instances will be hosting different revisions
66 –

Deployment Methods

Method	Impact of Failed Deployment	Deploy Time	Zero Downtime	No DNS Change	Rollback Process	Code Deployed To
All at once	Downtime	◷	x	✓	Redeploy	Existing instances
Rolling	Single batch out of service; any successful batches prior to failure running new application version	◷ ◷ †	✓	✓	Redeploy	Existing instances
Rolling with additional batch	Minimal if first batch fails, otherwise, similar to **Rolling**	◷ ◷ ◷ †	✓	✓	Redeploy	New and existing instances
Immutable	Minimal	◷ ◷ ◷ ◷	✓	✓	Redeploy	New instances
Blue/green	Minimal	◷ ◷ ◷ ◷	✓	x	Swap URL	New instances

67 – Termination hook for ASG and CW Events rule to trigger a Lambda that invokes an SSM Run Command to send log files from EC2 to S3; Create Lambda triggered by S3 event for PUT to write to DDB



68 – You can use .**ebextensions** w/ your web app's source code to configure your env and customize the AWS resources it contains – you can use the **option_settings** key to modify the **env config** ie: change config of ALB for HTTP to HTTPS redirection rule for EB
69 – Step functions useful for lambdas CW events and **copying EBS snapshots to other regions** and **handling errors** https://aws.amazon.com/blogs/compute/automating-amazon-ebs-snapshot-management-with-aws-step-functions-and-amazon-cloudwatch-events/
70 – DDB Lambda read from stream and pass payload to SNS, other lambdas read from SNS; **fan out** https://aws.amazon.com/blogs/database/how-to-perform-ordered-data-replication-between-applications-by-using-amazon-dynamodb-streams/

71 – Monitor and automatically react to changes in CodeDeploy using CW Alarms – can send notification to SNS, stop CD deployment, change state of instance, **automated rollback deployment**;
LambdaCanary10Percent10Minutes – 10% of new traffic for 10 minutes then all traffic shifted to new version
LambdaLinear10Percent10Minutes – 10% of new traffic in 10 minute increments until 100%
72 – Define commonly used architectures as CF templates, create **Service Catalog** stacks from these templates, ensure tagging done properly, place IAM users in beginner group and allow users to only launch stacks from Service Catalog while restricting write access to other services
73 – Separate CloudFormation stacks into individual, logical components that have dependencies on each other using Exports and Imports, can be version controlled
74 – Create an IAM Service Role for instances to be able to call AssumeRole on SSM service, generate activation code and activation id for on prem servers, use these creds to register your on prem servers, they will appear with prefix "**mi-**" while EC2 will be "i-"
75 – Lambda has a max concurrent execution of 1k while API Gateway is 10k; API Gateway integrates w/**Step Functions**; example – hotel 3 hr workflow w/manual verification needs to sustain 5k requests at same time

Practice Test (might be duplicates from full test)
1 – ASG Target Tracking scaling can use the following metrics: avg CPU, avg network in/out, ALB request count per target; scheduled action tells ASG to perform scaling at specified times
2 – To enable CT and Config in all accounts/regions create separate StackSets, create Config aggregator for centralized account to track compliance, create CW Event to generate events when compliance is breached and subscribe a Lambda to it that will send out notifications
3 – AMI for static, EC2 User Data script for dynamic part at runtime
4 – Lambda CodeDeploy small amount of traffic for 10 min then shift all – LambdaCanary10Percent10Minutes
Auto rollback if too many crashes – CW Alarm on Lambda CW Metrics and associate w/CD
5 – **Service Catalog** allows IT admins to create, manage and distribute catalogs of approved products to end users; ensure tagging is done properly and place IAM users in beginner group and allow the users to only launch stacks from Service Catalog
6 – CodeBuild stage can acquire ECR creds using CLI helpers or IAM role, build the image and push to ECR – upon success, start CodeDeploy stage w/target being ECS service
NOTE: CW Event Rule does NOT support CodeDeploy as a target
7 – ASG keeps terminating instances after booting up – set an instance into Standby right after it has launched; instances in Standby are part of ASG but do not actively handle traffic; hooks can work but they come with a one hour timeout
8 – ALB w/ASG and health check having issues connecting to DB – solution to enhance health check so return status code corresponds to connectivity of db