Gruntwork.io

Production Readiness Checklist

Are you ready to go to prod on AWS? Use this checklist to find out.

Everything you need to do before you go live

This checklist is your guide to the best practices for deploying secure, scalable, and highly available infrastructure in AWS. Before you go live, go through each item, and make sure you haven't missed anything important!

1. Server-side
2. Client-side
3. Data storage
4. Scalability and High Availability
5. Continuous Integration
6. Continuous Delivery
7. Networking
8. Security
9. Monitoring
10. Cost optimization

Your checklist state will be saved to Local Storage. ☑ ☐ ↕ ↕ 🖶

Server-side

- ☐ Build AMIs

  If you want to run your apps directly on EC2 Instances, you shoul… [more]

- [ ] Deploy AMIs using Auto Scaling Groups
  The best way to deploy an AMI is typically to run it as an Auto Sca… [more]

- [ ] Build Docker images
  If want to run your apps as containers, you should package your … [more]

- [ ] Deploy Docker images using ECS, EKS, or Fargate
  You have several options for running Docker containers in AWS. … [more]

- [ ] Deploy serverless apps using Lambda and API Gateway
  If you want to build serverless apps, you should package them as… [more]

- [ ] Configure CPU, memory, and GC settings
  Configure CPU settings (e.g., ensure your app uses all available C… [more]

- [ ] Configure hard drives
  Configure the root volume (https://docs.aws.amazon.com/AWSEC2/latest/User… [more]

Client-side

- [ ] Pick a JavaScript framework
  If you are building client-side applications in the browser, you m… [more]

- [ ] Pick a compile-to-JS language
  JavaScript has a number of problems and limitations, so you ma… [more]

- [ ] Pick a compile-to-CSS language
  CSS has a number of problems and limitations, so you may wish … [more]

- [ ] Optimize your assets
  All CSS and JavaScript should be minified and all images should … [more]

- [ ] Use a static content server
  You should serve all your static content (CSS, JS, images, fonts) fr… [more]

- [ ] Use a CDN
  Use CloudFront (https://aws.amazon.com/cloudfront/) as a Content Distrib… [more]

- [ ] Configure caching
  Think carefully about versioning, caching, and cache-busting for … [more]

Data storage

- [ ] Deploy relational databases
  Use Amazon's Relational Database Service (RDS) (https://aws.amazon.c… [more]

- [ ] Deploy NoSQL databases
  Use Elasticache (https://aws.amazon.com/elasticache/) if you want to use R… [more]

- [ ] Deploy queues
  Use Amazon Simple Queue Service (SQS) (https://aws.amazon.com/sqs/) … [more]

- [ ] Deploy search tools
  Use Amazon Elasticsearch (https://aws.amazon.com/elasticsearch-service/) or … [more]

- [ ] Deploy stream processing tools
  Use Amazon Managed Streaming for Apache Kafka (MSK) (https://a… [more]

- [ ] Deploy a data warehouse
  Use Amazon Redshift (https://aws.amazon.com/redshift/) for data warehou… [more]

- [ ] Deploy big data systems
  Use Amazon EMR (https://aws.amazon.com/emr/) to run Hadoop, Spark, … [more]

- [ ] Set up cron jobs
  Use AWS Lambda Scheduled Events (https://docs.aws.amazon.com/lambda… [more]

- [ ] Configure disk space
  Configure enough disk space on your system for all the data you … [more]

- [ ] Configure backup
  Configure backup for all of your data stores. Most Amazon-mana… [more]

- [ ] Configure cross-account backup
  Copy all of your backups to a separate AWS account for extra red… [more]

- [ ] Test your backups
  If you never test your backups, they probably don't work. Create … [more]

- [ ] Set up schema management
  For data stores that use a schema, such as relational databases, … [more]

Scalability and High Availability

- [ ] Choose between a Monolith and Microservices
  Ignore the hype and stick with a monolithic architecture as long … [more]

- [ ] Configure service discovery
  If you do go with microservices, one of the problems you'll need … [more]

- [ ] Use multiple Instances
  Always run more than one copy (i.e., more than one EC2 Instance… [more]

- [ ] Use multiple Availability Zones
  Configure your Auto Scaling Groups (https://docs.aws.amazon.com/autosca… [more]

- [ ] Set up load balancing
  Distribute load across your apps and Availability Zones using Am… [more]

- [ ] Use Auto Scaling
  Use auto scaling (https://aws.amazon.com/autoscaling/) to automatically sc… [more]

- [ ] Configure Auto Recovery
  Configure a process supervisor such as systemd (https://github.com/sy… [more]

- [ ] Configure graceful degradation
  Handle failures in your dependencies (e.g., a service not respond… [more]

- [ ] Perform load tests and use chaos engineering
  Run load tests against your infrastructure to figure out when it fal… [more]

Continuous Integration

- [ ] Pick a Version Control System
  Check all code into a Version Control System (VCS). The most po… [more]

- [ ] Do code reviews
  Set up a code review process in your team to ensure all commits … [more]

- [ ] Configure a build system
  Set up a build system for your project, such as Gradle (https://gradle.… [more]

- [ ] Use dependency management
  Your build systems should allow you to explicitly define all the of … [more]

- [ ] Configure static analysis
  Configure your build system so it can run static analysis tools (http… [more]

- [ ] Set up automatic code formatting
  Configure your build system to automatically format the code ac… [more]

- [ ] Set up automated tests
  Configure your build system so it can run automated tests on you… [more]

- [ ] Publish versioned artifacts
  Configure your build system so it can package your app into a de… [more]

- [ ] Set up a build server
  Set up a server to automatically run builds, static analysis, auto… [more]

Continuous Delivery

- [ ] Create deployment environments
  Define separate "environments" such as dev, stage, and prod. Ea… [more]

- [ ] Set up per-environment configuration
  Your apps may need different configuration settings in each envir… [more]

- [ ] Define your infrastructure as code
  Do not deploy anything by hand, by using the AWS Console, or th… [more]

- [ ] Test your infrastructure code
  If all of your infrastructure is defined as code, you can create aut… [more]

- [ ] Set up immutable infrastructure
  Don't update EC2 Instance or Docker containers in place. Instead,… [more]

- [ ] Promote artifacts
  Deploy immutable artifacts to one environment at a time, and pr… [more]

- [ ] Roll back in case of failure
  If you use immutable, versioned artifacts as your unit of deploym… [more]

- [ ] Automate your deployments
  One of the advantages of defining your entire infrastructure as co… [more]

- [ ] Do zero-downtime deployments
  There are several strategies you can use for Zero-downtime depl… [more]

- [ ] Use canary deployments
  Instead of deploying the new version of your code to all servers, … [more]

- [ ] Use feature toggles
  Wrap all new functionality in an if-statement that only evaluates … [more]

Networking

- [ ] Set up VPCs
  Don't use the Default VPC, as everything in it is publicly accessibl… [more]

- [ ] Set up subnets
  Create three "tiers" of subnets (https://docs.aws.amazon.com/AmazonVPC/lat… [more]

- [ ] Configure Network ACLs
  Create Network Access Control Lists (NACLs) (https://docs.aws.amazon.c… [more]

- [ ] Configure Security Groups
  Every AWS resource (e.g., EC2 Instances, Load Balancers, RDS DB… [more]

- [ ] Configure Static IPs
  By default, all AWS resources (e.g., EC2 Instances, Load Balancers… [more]

- [ ] Configure DNS using Route 53
  Manage DNS entries using Route 53 (https://aws.amazon.com/route53/). Y… [more]

Security

- [ ] Configure encryption in transit
  Encrypt all network connections using TLS (https://en.wikipedia.org/wiki… [more]

- [ ] Configure encryption at rest
  Encrypt the root volume of each EC2 Instance by using the encry… [more]

- [ ] Set up SSH access
  Do NOT share EC2 KeyPairs (https://docs.aws.amazon.com/AWSEC2/latest/User… [more]

- [ ] Deploy a Bastion Host
  Just about all your EC2 Instances should be in private subnets an… [more]

- [ ] Deploy a VPN Server
  We typically recommend running a VPN Server as the entry point… [more]

- [ ] Set up a secrets management solution
  NEVER store secrets in plaintext. Developers should store their se… [more]

- [ ] Use server hardening practices
  Every server should be hardened to protect against attackers. Thi… [more]

- [ ] Go through the OWASP Top 10
  Browse through the Top 10 Application Security Risks (https://www.o… [more]

- [ ] Go through a security audit
  Have a third party security service perform a security audit and d… [more]

- [ ] Sign up for security advisories
  Join the security advisory mailing lists for any software you use a… [more]

- [ ] Create IAM Users
  Create an IAM User (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users… [more]

- [ ] Create IAM Groups
  Manage permissions for IAM users using IAM Groups (https://docs.aws… [more]

- [ ] Create IAM Roles
  Give your AWS resources (e.g., EC2 Instances, Lambda Functions)… [more]

- [ ] Create cross-account IAM Roles
  If you are using multiple AWS accounts (e.g., one for dev and one … [more]

- [ ] Create a password policy and enforce MFA
  Set a password policy (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_cr… [more]

- [ ] Record audit Logs
  Enable CloudTrail (https://aws.amazon.com/cloudtrail/) to maintain an aud… [more]

Monitoring

- [ ] Track availability metrics
  The most basic set of metrics: can a user access your product or … [more]

- [ ] Track business metrics
  Metrics around what users are doing with your product, such as … [more]

- [ ] Track application metrics
  Metrics around what your application is doing, such as QPS, late… [more]

- [ ] Track server metrics
  Metrics around what your hardware is doing, such as CPU, memo… [more]

- [ ] Configure services for observability
  Record events and stream data from all services. Slice and dice it … [more]

- [ ] Store logs
  To prevent log files from taking up too much disk space, configur… [more]

- [ ] Set up alerts
  Configure alerts when critical metrics cross pre-defined threshol… [more]

Cost optimization

- [ ] Pick proper EC2 Instance types and sizes
  AWS offers a number of different Instance Types (https://aws.amazon.c… [more]

- [ ] Use Spot EC2 Instances for background jobs
  EC2 Spot Instances (https://aws.amazon.com/ec2/spot/) allow you to "bid" … [more]

- [ ] Use Reserved EC2 Instances for dedicated work
  EC2 Reserved Instances (https://aws.amazon.com/ec2/pricing/reserved-instance… [more]

- [ ] Shut down EC2 Instances and RDS DBs when not using them
  You can shut down (but not terminate!) EC2 Instances and RDS D… [more]

- [ ] Use Auto Scaling
  Use Auto Scaling (https://aws.amazon.com/autoscaling/) to increase the nu… [more]

- [ ] Use Docker when possible
  If you deploy everything as an AMI directly on your EC2 Instances… [more]

- [ ] Use Lambda when possible
  For all short (15 min or less) background jobs, cron jobs, ETL jobs… [more]

- [ ] Clean up old data with S3 Lifecycle settings
  If you have a lot of data in S3, make sure to take advantage of S3 … [more]

- [ ] Clean up unused resources
  Use tools such as cloud-nuke (https://github.com/gruntwork-io/cloud-nuke) a… [more]

- [ ] Learn to analyze your AWS bill
  Learn to use tools such as Cost and Usage Report (https://docs.aws.am… [more]

- [ ] Create billing alarms
  Create billing alerts (https://docs.aws.amazon.com/AmazonCloudWatch/latest/mo… [more]

Don't panic

Creating a production-ready infrastructure is a lot of work. But don't worry, we've got you covered. Just about everything in this checklist is already part of the Gruntwork Infrastructure as Code Library (/infrastructure-as-code-library/) and can be deployed in your AWS account in 1 day as part of the Reference Architecture (/reference-architecture/).