Configuration Manual: qpAdm AWS Benchmarking Pipeline

MSc Research Project – Cloud Computing

Utkarsh Singh (ID: 23332140)

National College of Ireland

Executive Summary

This manual describes how to deploy, run, and reproduce the qpAdm/qpfstats benchmarking pipeline used in my MSc project, "Performance Optimization and Cost Analysis of qpAdm Genomic Modeling on AWS Cloud Infrastructure" (NCI, Cloud Computing). It is written as a practical runbook: you can follow it to recreate the same infrastructure, build/pull the same container image, stage the same dataset layouts in S3, execute benchmark runs on Graviton EC2 workers, and regenerate the analysis outputs and figures.

The pipeline benchmarks qpAdm-style workflows across instance families (bench-c7g, bench-m7g, bench-r7g) and datasets (SGDP, 1000 Genomes chr22 SAS, AADR v62.0 thin/full variants). Compute is treated as ephemeral: workers boot from Auto Scaling Groups, stage inputs from S3 to local storage, run the containerised workflow, push run artifacts (logs, parameters, metrics.json) back to S3, and then scale back down. The accompanying analysis toolkit aggregates metrics.json and produces the CSV/JSON summaries and plots included in the submitted report.

Key features:

- Containerised workflow (multi-arch Docker image for arm64/amd64; version-pinned tools)
- Infrastructure-as-code (Terraform: IAM role + instance profile, security group, ASGs, log group)
- S3-first data layout (datasets/models/results/workdirs cached in S3; local staging to /data and /results)
- Auto-scaling ephemeral workers (cloud-init bootstrap + one-shot benchmark run + scale-in)
- Reproducible artifacts (metrics.json per run; aggregated outputs: analysis_ready.csv, failure_rates.csv, effect_sizes.csv, plots/)

Artifact bundle (analysis outputs)

If you already have the generated analysis bundle (out/), it should contain the following key files. These are produced by the Python analysis scripts described in Section 9.

- analysis_ready.csv — Main joined table used for figures/tables in the report.
- data_quality.json — Missingness / schema checks and row/column coverage summary.

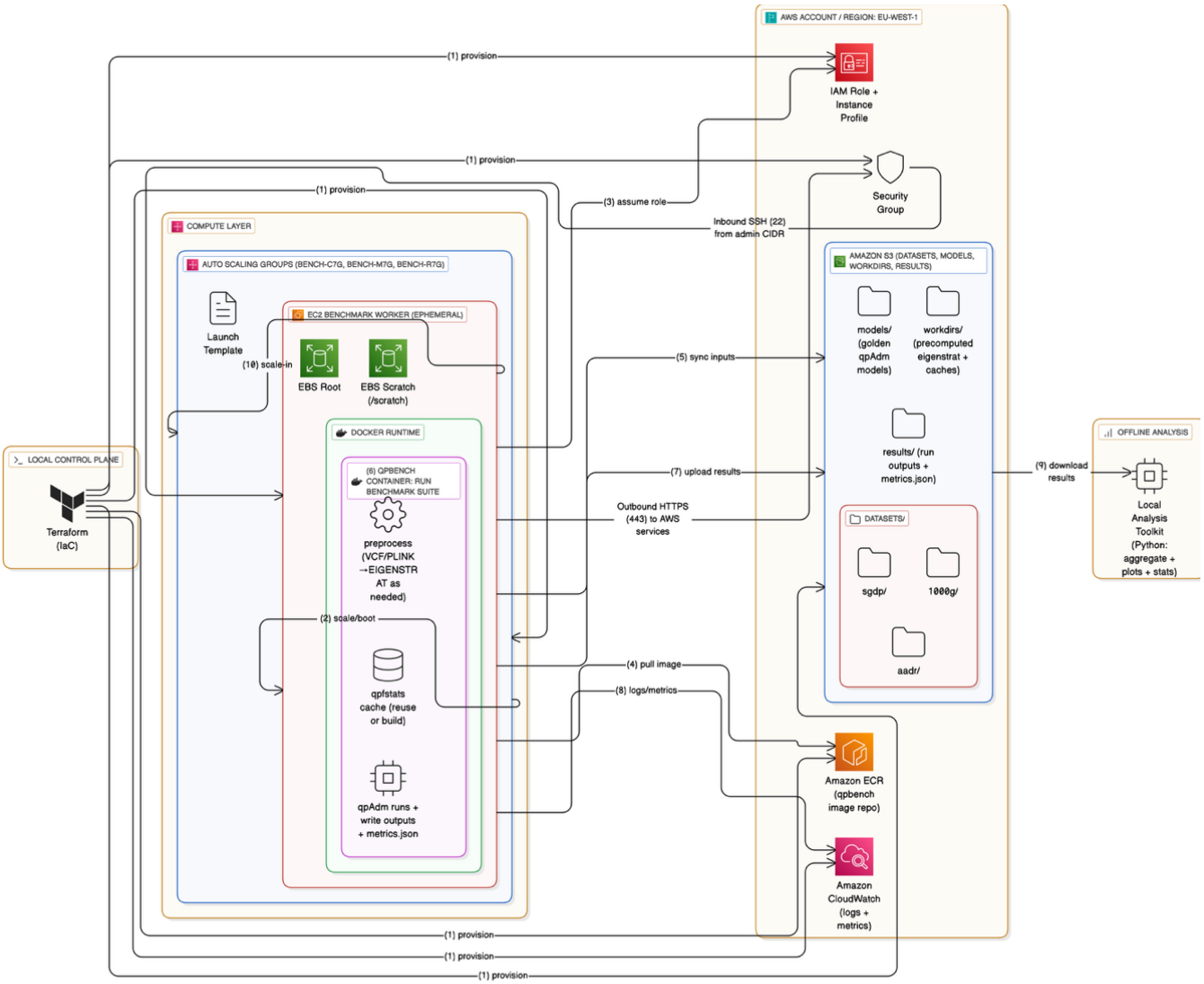
- failure_rates.csv — Failure rates by dataset/worker/preprocessing mode (including OOM).
- effect_sizes.csv — Effect size estimates for key comparisons (e.g., instance family differences).
- group_medians_runtime_cost.csv — Grouped medians for runtime and estimated cost.
- pareto_frontier.csv / pareto_frontier.png — Cost–runtime frontier points for configurations.
- plots/box_runtime_by_instance_family.png — Runtime distributions by instance family.
- plots/box_runtime_by_dataset.png — Runtime distributions by dataset.
- plots/scatter_runtime_vs_cost.png — Scatter of runtime vs cost (all configurations).
- plots/oom_rate_by_dataset_instance_family.png — OOM rates by dataset × instance family.
- plots/stage_breakdown_top_groups.png — Stage-level runtime breakdown for top groups.

Table of Contents

1. System Architecture Overview

1.1 High-Level Architecture

The pipeline separates durable storage (S3) from ephemeral computation (EC2):



1.2 Component Responsibilities

Component	Purpose	Key files / examples
Terraform	Provision ASGs, IAM, security groups, logging	terraform/main.tf, terraform/variables.tf, terraform/terraform.tfvars
Docker Image	Encapsulate qpAdm + dependencies	docker/Dockerfile, smoke/run_smoke_core.sh
User-Data Script	Bootstrap EC2 workers	terraform/user_data.sh.tpl
Golden Models	Define qpAdm left/right populations	config/models.yaml, smoke/build_golden_from_models.py
Analysis Toolkit	Aggregate results, generate stats/plots	analysis/aggregate_results.py, analysis/run_analysis.py, analysis/plot_results.py

1.3 Workflow Execution Flow

1. Terraform provisions infrastructure.
2. Docker image is built locally and pushed to Amazon ECR.

3. Datasets and golden models are stored in Amazon S3.
4. Auto Scaling Group desired capacity is set to 1 for a worker class.
5. EC2 worker bootstraps via user-data, stages data from S3, pulls container image, runs benchmarks, and uploads results.
6. Worker scales the Auto Scaling Group back to 0 after completion.
7. Results are downloaded locally for offline analysis.

2. Prerequisites and Requirements

2.1 Local Software Requirements

Tool	Minimum version	Purpose
AWS CLI	v2.x	Interact with AWS services
Terraform	≥ 1.5.0	Infrastructure provisioning
Docker	≥ 24.0 (with buildx)	Multi-arch container builds
Git	Any recent	Version control
Python	≥ 3.10	Offline analysis toolkit
jq (optional)	Any recent	JSON manipulation

Installation commands (Ubuntu/Debian):

```
# AWS CLI
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

# Terraform
wget https://releases.hashicorp.com/terraform/1.5.7/terraform_1.5.7_linux_amd64.zip
unzip terraform_1.5.7_linux_amd64.zip
sudo mv terraform /usr/local/bin/

# Docker (with buildx)
sudo apt-get update
sudo apt-get install -y docker.io docker-buildx

# Python + pip
sudo apt-get install -y python3 python3-pip python3-venv

# jq
sudo apt-get install -y jq
```

2.2 AWS Account Requirements

Required AWS services:

- Amazon EC2 (Graviton3 instances: c7g, m7g, r7g)
- Auto Scaling
- Amazon S3
- Amazon ECR
- Amazon CloudWatch
- AWS IAM

Region: eu-west-1 (Ireland) is used throughout this manual.

Account quotas: ensure capacity for at least 3 concurrent instances and associated Auto Scaling groups.

2.3 IAM Permissions Required

Terraform execution (local user/role) requires permissions to create/modify EC2, Auto Scaling, IAM, ECR, S3 and CloudWatch resources.

Example high-level policy (tighten for production):

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
    "Effect": "Allow",
    "Action": [
     "ec2:*",
     "autoscaling:*",
     "iam:CreateRole",
     "iam:AttachRolePolicy",
     "iam:CreateInstanceProfile",
     "iam:AddRoleToInstanceProfile",
     "iam:PassRole",
     "ecr:CreateRepository",
     "ecr:PutLifecyclePolicy",
     "s3:CreateBucket",
     "s3:PutBucketVersioning",
     "s3:PutEncryptionConfiguration",
     "logs:CreateLogGroup",
     "logs:PutRetentionPolicy"
    ],
    "Resource": "*"
   }
 ]
}
```

EC2 instance profile (created by Terraform) must allow:

- ECR: GetAuthorizationToken, BatchGetImage (and related pull actions)
- S3: GetObject/PutObject (and ListBucket as required) for the data bucket/prefixes
- CloudWatch Logs: CreateLogStream, PutLogEvents
- Auto Scaling: SetDesiredCapacity (to scale itself down)
- CloudWatch metrics: PutMetricData (if emitting custom metrics)

2.4 Network Requirements

Security Group rules:

- Inbound: SSH (TCP/22) from an admin CIDR (preferably your IP /32).
- Outbound: HTTPS (TCP/443) to AWS APIs and public endpoints required for package installs.

VPC: uses the default VPC in eu-west-1 unless modified in Terraform.

3. Initial Setup

3.1 Clone Repository

```
git clone https://github.com/UT07/Performance-Optimization-and-Cost-Analysis-of-qpAdm-Genomic-Modeling.git
cd qpbench
```

Expected directory structure:

```
repo-root/
  terraform/              # Terraform (ASGs, IAM, SG, ECR, S3 wiring)
  docker/              # Container build context (Dockerfile)
  smoke/               # Runner scripts (qpAdm + preprocess + suite)
  config/             # models.yaml, cost assumptions
  analysis/            # aggregate + plots + stats
  README.md
```

3.2 Configure AWS Credentials

Option A: Named profile

```
aws configure --profile qpbench
export AWS_PROFILE=qpbench
export AWS_REGION=eu-west-1
```

Verify:

```
aws sts get-caller-identity
```

3.3 Create S3 Bucket

```
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=eu-west-1
BUCKET_NAME="qpbench-datasets-${ACCOUNT_ID}-${REGION}"

aws s3 mb s3://${BUCKET_NAME} --region ${REGION}
aws s3api put-bucket-versioning --bucket ${BUCKET_NAME} --versioning-configuration Status=Enabled
```

Create required prefixes (optional, but helpful for organisation):

```
aws s3api put-object --bucket ${BUCKET_NAME} --key datasets/
aws s3api put-object --bucket ${BUCKET_NAME} --key golden/
aws s3api put-object --bucket ${BUCKET_NAME} --key results/
aws s3api put-object --bucket ${BUCKET_NAME} --key cache/
```

3.4 Create ECR Repository

```
aws ecr create-repository   --repository-name qpbench   --region eu-west-1   --image-scanning-configuration scanOnPush=true
```

3.5 Generate/Import an SSH Key Pair

```
aws ec2 create-key-pair   --key-name qpadm   --region eu-west-1   --query 'KeyMaterial'   --output text > ~/.ssh/qpadm.pem

chmod 400 ~/.ssh/qpadm.pem
```

4. Infrastructure Deployment

4.1 Configure Terraform Variables

Project-specific values used in my runs

In the submitted experiments I kept the region fixed to eu-west-1 and used one Auto Scaling Group per worker class. The common baseline was c7g.xlarge (bench-c7g) and m7g.xlarge (bench-m7g). For AADR-heavy runs, r7g.2xlarge (bench-r7g) was used after observing OOM events on smaller memory sizes during qpfstats. The example terraform.tfvars in this manual mirrors that setup; you only need to substitute your account ID, bucket name, and SSH CIDR.

```
cd infra
cp terraform.tfvars.example terraform.tfvars
nano terraform.tfvars
```

Example terraform.tfvars:

```
aws_account_id = "539518671472"
region         = "eu-west-1"
data_bucket    = "qpbench-datasets-539518671472-eu-west-1"
ecr_repo       = "qpbench"
image_tag      = "dev"
ssh_key_name   = "qpadm"
ssh_cidr       = "203.0.113.0/32"

# Ubuntu 22.04 ARM64 AMI for eu-west-1 (check for latest)
ami_id = "ami-033777517dfcfb37b"

workers = [
  {
    name              = "bench-c7g"
    instance_type     = "c7g.xlarge"
    volume_size_gb    = 150
    data_volume_size_gb = 150
    min_size          = 0
    max_size          = 1
    desired_capacity  = 0
  },
  {
    name              = "bench-m7g"
    instance_type     = "m7g.xlarge"
    volume_size_gb    = 150
    data_volume_size_gb = 150
    min_size          = 0
    max_size          = 1
    desired_capacity  = 0
  },
  {
    name              = "bench-r7g"
    instance_type     = "r7g.2xlarge"
    volume_size_gb    = 150
    data_volume_size_gb = 250
    min_size          = 0
    max_size          = 1
    desired_capacity  = 0
  }
]
```

Find the latest Ubuntu 22.04 ARM64 AMI:

```
aws ec2 describe-images   --region eu-west-1   --owners 099720109477   --filters
"Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-arm64-server-*"   --query 'Images | sort_by(@,
&CreationDate) | [-1].[ImageId,Name]'   --output text
```

4.2 Initialise Terraform

```
terraform init
```

4.3 Plan Infrastructure

```
terraform plan -var-file=terraform.tfvars -out=tfplan
```

4.4 Apply Infrastructure

```
terraform apply tfplan
```

Verify Auto Scaling groups:

```
aws autoscaling describe-auto-scaling-groups   --region eu-west-1   --query
'AutoScalingGroups[?contains(AutoScalingGroupName, `qpbench`)].AutoScalingGroupName'
```

5. Container Build and Publishing

5.1 Build Multi-Architecture Image

The Docker image includes ADMIXTOOLS (qpAdm, qpfstats, convertf), PLINK/PLINK2, bcftools/samtools/tabix and Python utilities.

```
ACCOUNT  ID=$(aws sts get-caller-identity --query Account --output text)
REGION=eu-west-1
IMAGE  TAG=dev

# Authenticate to ECR
aws ecr get-login-password --region ${REGION} |   docker login --username AWS --password-stdin
${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com

# Build and push multi-arch image
docker buildx build   --platform linux/arm64,linux/amd64   --file docker/Dockerfile   --tag
${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/qpbench:${IMAGE_TAG}   --push   .
```

If buildx is not enabled:

```
docker buildx create --name multiarch --use
docker buildx inspect --bootstrap
```

5.2 Verify Image Push

```
aws ecr describe-images   --repository-name qpbench   --region eu-west-1
```

5.3 Tag a Release (Optional)

```
GIT_SHA=$(git rev-parse --short HEAD)
docker buildx imagetools create   --tag
${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/qpbench:${GIT_SHA}
${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/qpbench:${IMAGE_TAG}
```

6. Data Preparation and Upload

6.1 Generate Golden Models

Golden models define qpAdm left/right population lists.

```
python3 smoke/gen_golden_models.py  --datasets aadr sgdp 1000G  --ind-aadr
"./Datasets/aadr/v62.0/thin_step4/v62.0_1240k_public.ind"  --ind-sgdp  "./Datasets/sgdp/eigenstrat/sgdp_tiny.ind"  --
ind-1000g "./Datasets/1000G/eigenstrat/1kg_chr22_sas.ind"  --max-models-aadr 200  --max-models-sgdp 60  --max-
models-1000g 140  --max-right 32  --out config/models.yaml
```

Build golden directories and sync to S3 (example):

```
python3 smoke/build_golden_from_models.py  config/models.yaml  golden/  --s3-prefix
s3://${BUCKET_NAME}/golden  --prune-s3  --exact-timestamps
```

6.2 Upload Datasets to S3

Example local layout:

```
Datasets/
├── sgdp/
│   └── eigenstrat/
│       ├── sgdp_tiny.ind
│       ├── sgdp  tiny.snp
│       └── sgdp_tiny.geno
├── 1000G/
│   ├── vcf/
│   │   └── chr22  sas.vcf.gz
│   └── eigenstrat/
│       └── 1kg_chr22_sas.{ind,snp,geno}
└── aadr/
    └── v62.0/
        └── thin_step4/
            ├── v62.0_1240k_public.ind
            ├── v62.0_1240k_public.snp
            └── v62.0_1240k_public.geno
```

Sync to S3 (examples):

```
aws s3 sync Datasets/sgdp/  s3://${BUCKET_NAME}/sgdp/  --exact-timestamps
aws s3 sync Datasets/1000G/ s3://${BUCKET_NAME}/1000G/ --exact-timestamps
aws s3 sync Datasets/aadr/  s3://${BUCKET_NAME}/aadr/  --exact-timestamps
```

AADR thinning example (convertf):

```
# Example: thin to every 4th SNP
convertf -p aadr_thin.par

# aadr_thin.par (example)
genotypename: v62.0_1240k_public.geno
snpname:      v62.0_1240k_public.snp
indivname:    v62.0_1240k_public.ind
outputformat: PACKEDANCESTRYMAP
genotypeoutname: v62.0_1240k_public_thin4.geno
snpoutname:      v62.0_1240k_public_thin4.snp
indivoutname:    v62.0_1240k_public_thin4.ind
```

6.3 Upload Precomputed Caches (Optional)

To speed repeated runs, qpfstats caches can be generated and uploaded.

```
qpfstats -p qpf.par
aws s3 cp sgdp_tiny_cache.qpf.txt s3://${BUCKET_NAME}/cache/
```

7. Running Benchmarks

7.1 Launch a Single Worker

Scale an Auto Scaling group to desired_capacity=1:

```
aws autoscaling set-desired-capacity  --region eu-west-1  --auto-scaling-group-name qpbench-bench-c7g-asg  --desired-capacity 1  --honor-cooldown
```

Monitor instance launch:

```
aws ec2 describe-instances  --region eu-west-1  --filters "Name=tag:WorkerClass,Values=bench-c7g"  --query 'Reservations[*].Instances[*].[InstanceId,State.Name,PublicIpAddress]'  --output table
```

7.2 Launch Multiple Workers (Parallel)

```
for asg in qpbench-bench-c7g-asg qpbench-bench-m7g-asg qpbench-bench-r7g-asg; do
  aws autoscaling set-desired-capacity  --region eu-west-1  --auto-scaling-group-name ${asg}  --desired-capacity 1 --honor-cooldown &
done
```

Note: for controlled benchmarking, run sequentially to reduce S3 contention and variability.

7.3 SSH to Worker (Debugging)

```
INSTANCE_IP=$(aws ec2 describe-instances  --region eu-west-1  --filters "Name=tag:WorkerClass,Values=bench-c7g" "Name=instance-state-name,Values=running"  --query 'Reservations[0].Instances[0].PublicIpAddress'  --output text)

ssh -i ~/.ssh/qpadm.pem ubuntu@${INSTANCE_IP}
```

On the instance:

```
sudo tail -f /var/log/cloud-init-output.log
sudo docker ps
sudo docker logs $(sudo docker ps -q)
ls -lh /results/
```

7.4 Manual Run (Local Testing)

```
docker pull ${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/qpbench:${IMAGE_TAG}

docker run --rm  -v $(pwd)/Datasets/sgdp/eigenstrat:/data  -v $(pwd)/golden:/opt/golden  -v $(pwd)/results_local:/results  -e GOLDEN=sgdp_French__Spanish  -e EIGENSTRAT_DIR=/data  -e INPUT_FORMAT=EIGENSTRAT  -e IND_BASENAME=sgdp_tiny ${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/qpbench:${IMAGE_TAG}  /opt/smoke/run_smoke_core.sh
```

8. Monitoring and Logs

8.1 CloudWatch Logs

List log groups:

```
aws logs describe-log-groups  --region eu-west-1  --log-group-name-prefix /qpbench/
```

Tail logs:

```
aws logs tail /qpbench/qpbench --follow --filter-pattern "stage" --region eu-west-1
```

8.2 CloudWatch Metrics (Examples)

CPU I/O wait can indicate preprocessing bottlenecks; memory usage is relevant for AADR-scale panels.

```
aws cloudwatch get-metric-statistics  --namespace CWAgent  --metric-name cpu_usage_iowait  --dimensions Name=AutoScalingGroupName,Value=qpbench-bench-c7g-asg  --start-time $(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%S)  --end-time $(date -u +%Y-%m-%dT%H:%M:%S)  --period 300  --statistics Average  --region eu-west-1
aws cloudwatch get-metric-statistics  --namespace CWAgent  --metric-name mem_used_percent  --dimensions Name=AutoScalingGroupName,Value=qpbench-bench-r7g-asg  --start-time $(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%S)  --end-time $(date -u +%Y-%m-%dT%H:%M:%S)  --period 300  --statistics Maximum  --region eu-west-1
```

8.3 Check Run Status

```
aws s3 ls s3://${BUCKET_NAME}/results/bench-c7g/ --recursive | tail -20
```

A run is considered complete when metrics.json exists for the run_id.

9. Results Collection and Analysis

9.1 Sync Results Locally

```
mkdir -p results_local
aws s3 sync s3://${BUCKET_NAME}/results/ results_local/ --exact-timestamps
```

9.2 Aggregate results (analysis_ready.csv)

aggregate_results.py walks the downloaded results tree and builds a single analysis-ready table. It handles the project's results_batches layout, keeps the original S3 run_dir recorded in metrics.json, and also computes run_dir_local to parse local log files. The script also backfills identity columns (dataset, input_format, golden/model_id, seed) for missing-metrics/OOM rows, and can optionally parse stdout.log/stage.log to recover stage timings.

```
python3 analysis/aggregate_results.py \

  --results-batches-root ./results_batches \

  --out ./analysis/out/analysis_ready.csv \

  --fmt csv \

  --parse-logs \

  --infer-missing-metrics \

  --cost-assumptions ./config/cost_assumptions.yaml
```

Key outputs: analysis/out/analysis_ready.csv — consolidated run-level table (one row per run).

9.3 Run statistical analysis and generate plots

The analysis stage consumes analysis_ready.csv and produces summary tables (failure rates, medians, effect sizes) plus the figures used in the Evaluation chapter (e.g., stage breakdown, runtime vs cost, OOM rates).

- analysis/out/failure_rates.csv — failure + OOM rates grouped by dataset/worker/preprocessing.
- analysis/out/group_medians_runtime_cost.csv — grouped medians for runtime and estimated cost.
- analysis/out/effect_sizes.csv — effect sizes for key pairwise comparisons.
- analysis/out/pareto_frontier.csv + analysis/out/pareto_frontier.png — Pareto frontier (cost vs runtime).
- analysis/out/plots/ — PNG figures, including: stage_breakdown_top_groups.png, scatter_runtime_vs_cost.png, box_runtime_by_instance_family.png, box_runtime_by_dataset.png, oom_rate_by_dataset_instance_family.png, and box_cost_by_instance_family.png.

• effect_sizes.csv — pairwise effect sizes (Cohen's d) for runtime and cost comparisons.

• pareto_frontier.csv — cost–performance frontier points used in the Pareto plot.

• data_quality.json — column completeness and basic sanity checks.

Key figures (analysis/out/ or analysis/out/plots/):

• pareto_frontier.png — Pareto frontier (runtime vs cost).

• plots/scatter_runtime_vs_cost.png — scatter of runtime vs cost by configuration.

• plots/stage_breakdown_top_groups.png — stage breakdown for representative groups.

• plots/box_runtime_by_instance_family.png — runtime distribution by instance family.

• plots/box_cost_by_instance_family.png — cost distribution by instance family.

• plots/oom_rate_by_dataset_instance_family.png — OOM rates by dataset and instance family.

10. Cleanup and Teardown

10.1 Stop All Workers

```
for asg in qpbench-bench-c7g-asg qpbench-bench-m7g-asg qpbench-bench-r7g-asg; do
  aws autoscaling set-desired-capacity     --region eu-west-1     --auto-scaling-group-name ${asg}     --desired-capacity 0
done
```

10.2 Destroy Infrastructure

```
cd infra
terraform destroy -var-file=terraform.tfvars
```

Note: S3 buckets and ECR repositories are typically deleted manually.

10.3 Download/Archive Final Results

```
aws s3 sync s3://${BUCKET_NAME}/results/ ./results_archive/ --exact-timestamps
aws s3 sync s3://${BUCKET_NAME}/cache/   ./cache_archive/   --exact-timestamps
```

10.4 Delete S3 Bucket

```
# Delete all objects first
aws s3 rm s3://${BUCKET_NAME}/ --recursive

# Then delete the bucket
aws s3 rb s3://${BUCKET_NAME}
```

10.5 Delete ECR Repository

```
aws ecr delete-repository   --repository-name qpbench   --region eu-west-1   --force
```

10.6 Revoke SSH Key

```
aws ec2 delete-key-pair --key-name qpadm --region eu-west-1
rm -f ~/.ssh/qpadm.pem
```

Appendix A: File Reference

File	Purpose	Location
terraform/main.tf	Terraform main configuration	terraform/
terraform/variables.tf	Terraform input variables	terraform/
terraform/terraform.tfvars	User-specific values (gitignored)	terraform/
terraform/user_data.sh.tpl	EC2 bootstrap script template	terraform/
docker/Dockerfile	Container image definition	docker/
smoke/run_smoke_core.sh	qpAdm runner (single model)	smoke/
smoke/run_benchmark_suite.sh	Batch runner (suite)	smoke/
smoke/preprocess_convertf.sh	VCF/PLINK → EIGENSTRAT converters	smoke/
config/models.yaml	Golden model definitions	config/
config/cost_assumptions.yaml	Hourly rates by instance family	config/
analysis/aggregate_results.py	Aggregate per-run metrics/logs into an analysis-ready CSV (analysis_ready.csv)	analysis/
analysis/run_analysis.py	Statistical analysis	analysis/
analysis/plot_results.py	Plotting script	analysis/
analysis/out/analysis_ready.csv	Unified per-run table used for all statistics and	Generated locally

	figures	
analysis/out/data_quality.json	Data completeness summary (missing metrics, log coverage, empty columns)	Generated locally
analysis/out/failure_rates.csv	Failure/OOM rates grouped by dataset × instance family	Generated locally
analysis/out/group_medians_runtime_cost.csv	Median runtime/cost summaries by group (dataset × instance family × mode)	Generated locally
analysis/out/effect_sizes.csv	Effect sizes (Cohen's d) for key comparisons	Generated locally
analysis/out/pareto_frontier.csv	Pareto-optimal configurations for runtime vs cost	Generated locally
analysis/out/pareto_frontier.png	Pareto frontier figure for thesis/reporting	Generated locally
analysis/out/plots/*.png	Publication-ready plots (boxplots, scatter runtime vs cost, stage breakdown, OOM rate)	Generated locally

Appendix B: Environment Variables

B.1 User-Data Environment Variables (examples)

Variable	Example	Purpose
REGION	eu-west-1	AWS region
ACCOUNT	539518671472	AWS account ID
DATA_BUCKET	qpbench-datasets-…	S3 bucket name
ECR_REPO	qpbench	ECR repository
IMAGE_TAG	dev	Docker image tag
WORKER_CLASS	bench-c7g	Worker class identifier
SCRATCH_DEVICE_NAME	/dev/sdf	EBS scratch device name

B.2 Container Environment Variables (examples)

Variable	Example	Purpose
GOLDEN	sgdp_French__Spanish	qpAdm model ID
EIGENSTRAT_DIR	/data	Input data path
INPUT_FORMAT	EIGENSTRAT	Input format (EIGENSTRAT/PLINK/VCF)
IND_BASENAME	sgdp_tiny	Basename for .ind/.snp/.geno
BLGSIZE	0.001	Block jackknife size
USE_PACKED	NO	Convert to PACKEDANCESTRYMAP?
AADR_THIN_STEP	1	SNP thinning step (1 = no thin)

SEED	1	Random seed for replicates

Appendix C: Cost Estimation

The thesis uses scenario-based pricing assumptions for internal comparisons. Use AWS pricing pages for current rates.

Example on-demand rates used in the evaluation (illustrative):

Instance type	**vCPUs**	**RAM (GiB)**	**Hourly rate (USD)**	**Spot rate (est.)**
c7g.large	2	4	$0.072	$0.025
c7g.xlarge	4	8	$0.144	$0.050
m7g.large	2	8	$0.081	$0.028
m7g.xlarge	4	16	$0.162	$0.057
r7g.large	2	16	$0.100	$0.035
r7g.xlarge	4	32	$0.201	$0.070
r7g.2xlarge	8	64	$0.402	$0.141

Appendix D: Troubleshooting Decision Tree

```
Run failed?
│
├─ metrics.json missing?
│   ├─ YES → Check CloudWatch logs for errors
│   │        ├─ ECR auth failed? → Verify IAM role
│   │        ├─ S3 permission denied? → Check bucket policy
│   │        └─ Container crashed? → SSH to instance, check docker logs
│   └─ NO → Run completed, check results
│
├─ Runtime > expected?
│   ├─ High cpu_usage_iowait? → Preprocessing bottleneck
│   │                          → Use precomputed EIGENSTRAT
│   └─ Low CPU usage? → Instance undersized for dataset
│                       → Use larger instance (m7g → c7g.xlarge)
│
└─ OOM (out-of-memory)?
    └─ For AADR: use r7g.2xlarge or thin SNPs
```