ThinkPanel

Installation Guide

AWS Marketplace — BOX Program Deployment

Version 1.0 | June 2026

Audience: AWS Engineers

1. Overview

ThinkPanel is an AI-powered chat panel delivered as a set of Docker containers and AWS CloudFormation nested stacks. This guide walks an AWS engineer through deploying ThinkPanel into a customer AWS account from start to finish using the BOX program delivery package.

1.1 Architecture Summary

The deployment provisions the following resources in the customer account:

- VPC with two public and two private subnets across two Availability Zones
- NAT Gateway for outbound internet access from private subnets
- Application Load Balancer (ALB) in the public subnets
- Amazon ECS Fargate cluster running two services:
  - Backend — Python/FastAPI on port 8000 (internal only, via service discovery)
  - Frontend — React/nginx on port 3000 (exposed through the ALB)
- Amazon RDS PostgreSQL 15 in private subnets (encrypted, deletion-protected)
- Amazon Cognito User Pool for authentication
- AWS Secrets Manager for all credentials, encrypted with a per-stack KMS key
- SSM Parameter Store for non-secret configuration
- Auto Scaling policies on both ECS services (CPU target: 60%)

1.2 Nested Stack Structure

Parameter	Value / Description
main.yaml	Root stack — the only file deployed directly
networking.yaml	VPC, subnets, route tables, NAT Gateway
security.yaml	KMS key, Security Groups
cognito.yaml	Cognito User Pool and App Client
data.yaml	RDS instance, Secrets Manager secrets, SSM parameters
compute.yaml	ECS cluster, task definitions, services, ALB, Auto Scaling

2. Prerequisites

Verify every item below before starting the deployment. Missing prerequisites are the most common cause of failed deployments.

Requirement	Minimum	Notes
AWS CLI	v2.x	Configured with credentials that have AdministratorAccess or a scoped IAM policy covering CloudFormation, ECS, ECR, RDS, Cognito, Secrets Manager, SSM, KMS, IAM, VPC
Target AWS region	Any	All resources deploy to the region you specify. Choose a region that supports ECS Fargate, RDS, and Cognito.
ECR repositories	Exists	aithinkpanelbe and aithinkpanelfe must exist in the target account and region before deployment.
Docker images	Pushed	Both images must be pushed to the ECR repos above before deploying. See Section 3.
S3 bucket	Exists	A bucket in the target region to host the nested CloudFormation templates. Create one if needed.
ACM certificate	Optional	Required only for HTTPS. Must be in the same region. See Section 5.
IAM permissions	Required	The deploying principal needs CAPABILITY_NAMED_IAM — it creates IAM roles with explicit names.

3. Preparing Container Images

> **NOTE** *If the customer already has the images pushed to their ECR repositories, skip to Section 4.*

3.1 Authenticate Docker to ECR

Replace the placeholders with the customer's account ID and target region:

```
aws ecr get-login-password --region <REGION> \
  | docker login --username AWS \
  --password-stdin <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com
```

3.2 Create ECR Repositories (if they do not exist)

```
aws ecr create-repository --repository-name aithinkpanelbe --region <REGION>
aws ecr create-repository --repository-name aithinkpanelfe --region <REGION>
```

3.3 Tag and Push the Backend Image

The backend image is built from the root Dockerfile included in the delivery package:

```
# From the root of the delivery package
docker build --platform linux/amd64 \
  -t <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/aithinkpanelbe:<TAG> .

docker push <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/aithinkpanelbe:<TAG>
```

3.4 Build and Push the Frontend Image

> **IMPORTANT** *VITE_* variables are baked into the React bundle at build time, not at runtime. You must set VITE_API_BASE_URL to the ALB URL before building. Because the ALB URL is not known until after the CloudFormation deploy completes, follow the two-phase approach below.*

Phase 1 — Deploy infrastructure first (Section 4), then retrieve the ALB DNS:

```
ALB_URL=$(aws cloudformation describe-stacks \
  --stack-name <STACK_NAME> \
  --query "Stacks[0].Outputs[?OutputKey=='FrontendURL'].OutputValue" \
  --output text)
echo $ALB_URL
```

Phase 2 — Build and push the frontend with the real ALB URL:

```
cd thinkpanel-fe
docker build --platform linux/amd64 \
  --build-arg VITE_API_BASE_URL=${ALB_URL} \
  -t <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/aithinkpanelfe:<TAG> .
```

```
docker push <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/aithinkpanelfe:<TAG>
```

Phase 3 — Update the ECS service to pull the new frontend image:

```
aws ecs update-service \
  --cluster <STACK_NAME>-cluster \
  --service <STACK_NAME>-frontend \
  --force-new-deployment \
  --region <REGION>
```

4. Deploying CloudFormation

4.1 Upload Nested Templates to S3

All six YAML files must be in the S3 bucket before the root stack is deployed:

```
BUCKET=<YOUR_TEMPLATES_BUCKET>
REGION=<YOUR_REGION>

cd aithinkpanel-cloudformation

aws s3 cp networking.yaml s3://${BUCKET}/networking.yaml
aws s3 cp security.yaml   s3://${BUCKET}/security.yaml
aws s3 cp cognito.yaml    s3://${BUCKET}/cognito.yaml
aws s3 cp data.yaml       s3://${BUCKET}/data.yaml
aws s3 cp compute.yaml    s3://${BUCKET}/compute.yaml
```

> **NOTE** *Do not upload main.yaml to S3 — it is deployed directly from the local filesystem.*

4.2 Deploy the Root Stack

Run the following command, substituting all placeholder values:

```
aws cloudformation deploy \
  --template-file main.yaml \
  --stack-name <STACK_NAME> \
  --region ${REGION} \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides \
      TemplatesBucket=${BUCKET} \
      ImageTag=<TAG> \
      DBInstanceClass=db.t3.small \
      DBBackupRetentionDays=7
```

> **NOTE** *Stack deployment typically takes 15–20 minutes. RDS provisioning is the longest step.*

4.3 Parameters Reference

Parameter	Value / Description
TemplatesBucket	Required. Name of the S3 bucket holding the nested YAMLs.
ImageTag	Docker image tag to deploy. Use a pinned version (e.g. 1.0.0) in production, not latest.

BackendImageUri	Optional. Full ECR URI for the backend image. Auto-derived from account/region + ImageTag when left blank.
FrontendImageUri	Optional. Full ECR URI for the frontend image. Auto-derived from account/region + ImageTag when left blank.
DBInstanceClass	RDS instance class. Default: db.t3.small. Options: db.t3.micro, db.t3.small, db.t3.medium, db.r6g.large.
DBBackupRetentionDays	Number of days to retain automated RDS backups. Default: 7. Range: 1–35.
AcmCertificateArn	Optional. ACM certificate ARN for HTTPS. Leave blank for HTTP-only. See Section 5.

4.4 Verify the Deployment

Once the stack reaches CREATE_COMPLETE, retrieve the frontend URL:

```
aws cloudformation describe-stacks \
  --stack-name <STACK_NAME> \
  --query "Stacks[0].Outputs" \
  --output table
```

The FrontendURL output will be the ALB DNS name. Open it in a browser to confirm ThinkPanel is running.

5. Enabling HTTPS

HTTPS is optional at initial deployment. The customer issues and manages their own ACM certificate. Once available, HTTPS can be enabled by updating the stack.

5.1 Issue an ACM Certificate

The certificate must be in the same AWS region as the deployment. Issue it via the AWS Console (ACM > Request certificate) or CLI:

```
aws acm request-certificate \
  --domain-name <YOUR_DOMAIN> \
  --validation-method DNS \
  --region <REGION>
```

Complete DNS validation, then note the certificate ARN.

5.2 Update the Stack

```
aws cloudformation deploy \
  --template-file main.yaml \
  --stack-name <STACK_NAME> \
  --region <REGION> \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides \
      TemplatesBucket=${BUCKET} \
      ImageTag=<TAG> \
      AcmCertificateArn=arn:aws:acm:<REGION>:<ACCOUNT_ID>:certificate/<CERT_ID>
```

This update will: add an HTTPS listener on port 443, configure the TLS 1.3 security policy, and redirect all HTTP traffic on port 80 to HTTPS automatically.

6. Post-Deployment Steps

6.1 Update Cognito Callback URL

On the first deployment Cognito is configured with a localhost callback URL as a placeholder. After you have the ALB URL (and domain, if HTTPS is configured), update it:

1. Open the AWS Cognito console in the target region.
2. Navigate to User Pools > <STACK_NAME>-<ACCOUNT_ID>-auth-users > App clients.
3. Select the app client and click Edit hosted UI.
4. Replace http://localhost:3000 with the real ALB URL (e.g. https://your-domain.com).
5. Save changes.

Alternatively, redeploy the stack passing the ALBCallbackUrl parameter to the Cognito nested stack. Refer to the README in the cloudformation directory.

6.2 Create the First Admin User

ThinkPanel uses Cognito for authentication. Create the first admin user via CLI:

```
# Get the User Pool ID from stack outputs
POOL_ID=$(aws cloudformation describe-stacks \
  --stack-name <STACK_NAME>-Cognito \
  --query "Stacks[0].Outputs[?OutputKey=='UserPoolId'].OutputValue" \
  --output text)

aws cognito-idp admin-create-user \
  --user-pool-id ${POOL_ID} \
  --username admin@example.com \
  --temporary-password TempPass123! \
  --message-action SUPPRESS
```

6.3 Verify ECS Services are Healthy

```
aws ecs describe-services \
  --cluster <STACK_NAME>-cluster \
  --services <STACK_NAME>-backend <STACK_NAME>-frontend \
  --query
"services[*].{Name:serviceName,Running:runningCount,Desired:desiredCount,Status:status
}" \
  --output table
```

Both services should show runningCount = desiredCount = 1 and status = ACTIVE.

6.4 Verify RDS Connectivity

ECS tasks connect to RDS via the private security group. Connectivity is implicit if the backend service is running. To verify independently, use the RDS endpoint from SSM:

```
aws ssm get-parameter \
  --name /aithinkpanel/<STACK_NAME>/PLATFORM_PG_HOST \
  --query "Parameter.Value" --output text
```

7. Stack Teardown

> **IMPORTANT** *Deleting the stack will permanently delete all secrets, SSM parameters, and ECS resources. The RDS instance is protected by DeletionProtection and will NOT be deleted automatically — it must be manually disabled and deleted afterward.*

7.1 Disable RDS Deletion Protection

```
DB_ID=$(aws rds describe-db-instances \
  --query "DBInstances[?contains(DBInstanceIdentifier,
'<STACK_NAME>')].DBInstanceIdentifier" \
  --output text)

aws rds modify-db-instance \
  --db-instance-identifier ${DB_ID} \
  --no-deletion-protection \
  --apply-immediately
```

7.2 Delete the Stack

```
aws cloudformation delete-stack \
  --stack-name <STACK_NAME> \
  --region <REGION>

# Monitor progress
aws cloudformation wait stack-delete-complete --stack-name <STACK_NAME>
```

7.3 Delete the RDS Instance

```
aws rds delete-db-instance \
  --db-instance-identifier ${DB_ID} \
  --skip-final-snapshot
```

> **NOTE** *Remove the --skip-final-snapshot flag and add --final-db-snapshot-identifier <name> if you need a final backup before deletion.*

8. Troubleshooting

Parameter	Value / Description
Stack stuck in CREATE_IN_PROGRESS	Check the CloudFormation Events tab in the console. Most failures are in the Compute nested stack — look for ECS service stabilization errors. Confirm ECR images exist and are accessible from the target region/account.
ECS tasks failing to start	Check CloudWatch Logs at /ecs/<STACK_NAME>/backend or /ecs/<STACK_NAME>/frontend. Common causes: missing secrets (wrong ARN), ECR image pull failure, or RDS not yet accepting connections.
Frontend loads but API calls fail	VITE_API_BASE_URL is baked into the JS bundle. Confirm the frontend image was built with the correct ALB URL (Section 3.4). Check CORS_ORIGINS env var on the backend task matches the ALB URL exactly.
Cognito redirect_uri_mismatch error	The ALB callback URL has not been added to the Cognito app client. Follow Section 6.1 to update it.
RDS connection refused from ECS tasks	Verify the ECSServiceSG is the source in the RDSSG ingress rule. Check that both services are in PrivateSubnet (same AZ as RDS, or at least same VPC).
CloudFormation ROLLBACK on Cognito domain	The Cognito domain prefix must be globally unique. The stack name + account ID suffix should prevent this, but if you redeployed after a partial failure, the domain may still be reserved. Wait 5 minutes and retry, or change the stack name.

9. Security Notes

- All secrets are stored in AWS Secrets Manager, encrypted with a stack-specific KMS key with annual automatic key rotation enabled.
- IAM policies are scoped to the exact secret ARNs and SSM parameter paths created by this stack — no wildcard resource grants on Secrets Manager.
- RDS is deployed with PubliclyAccessible: false, DeletionProtection: true, and storage encrypted.
- ECS tasks run as a non-root user (appuser) inside the container.
- The backend service has no ALB attachment and is only reachable via internal service discovery (backend.aithinkpanel.local:8000).
- When AcmCertificateArn is provided, the ALB enforces TLS 1.3 (ELBSecurityPolicy-TLS13-1-2-2021-06) and redirects all HTTP traffic to HTTPS.
- The Cognito domain prefix includes the AWS account ID to guarantee global uniqueness and prevent domain squatting across customer deployments.

10. Quick Reference

10.1 Key Resource Name Pattern

All resources created by this stack are prefixed with the stack name, making them easy to identify:

Parameter	Value / Description
ECS Cluster	<STACK_NAME>-cluster
ECS Backend Service	<STACK_NAME>-backend
ECS Frontend Service	<STACK_NAME>-frontend
Execution IAM Role	<STACK_NAME>-ecs-execution-role
Task IAM Role	<STACK_NAME>-ecs-task-role
CloudWatch Log Group (BE)	/ecs/<STACK_NAME>/backend
CloudWatch Log Group (FE)	/ecs/<STACK_NAME>/frontend
Secrets Manager prefix	/aithinkpanel/<STACK_NAME>/
SSM Parameter prefix	/aithinkpanel/<STACK_NAME>/

10.2 Useful AWS Console Links

- CloudFormation Stacks: Console > CloudFormation > Stacks
- ECS Services: Console > ECS > Clusters > <STACK_NAME>-cluster
- CloudWatch Logs: Console > CloudWatch > Log groups > /ecs/<STACK_NAME>/
- Secrets Manager: Console > Secrets Manager > filter by /aithinkpanel/
- RDS: Console > RDS > Databases
- Cognito: Console > Cognito > User Pools