NETbuilder

labb

Cara for Pega

Customer Test Script. Prepared by Labb

August 11, 2026

Table of Contents

Overview

This script takes you end to end: subscribe, deploy, and confirm the deployment works. It assumes the prerequisites in DEPLOYMENT.md are already in place, namely a Pega OAuth client, a LiveKit project, and your provider API keys. Allow about 15 minutes, plus stack creation time.

What You Will Need Locally

- The AWS CLI, configured with credentials for the account you are deploying into, with permissions for CloudFormation, ECS, and SSM
- The Session Manager plugin for the AWS CLI, which the port-forwarding step requires
- A web browser

Step 0: Confirm Your AWS Account

Point the AWS CLI at the account you intend to deploy into. This is easy to get wrong if you work across multiple profiles or SSO sessions.

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

Check that the Account value in the output matches the one you expect before you continue. Every command in this script also accepts --region followed by your region, if that region is not already your CLI default.

Step 1: Subscribe

Cara for Pega is sold through a private offer, with quote-led pricing.

- You will receive an offer link by email
- Open the link and select Accept Offer in AWS Marketplace
- Accepting subscribes your AWS account and grants the entitlement that the deployed containers check at startup

Step 2: Launch the Stack

From the Marketplace listing page, or from the offer confirmation, open Launch CloudFormation Stack. The template loads with the Marketplace image URIs already filled in. Supply the remaining values:

- LiveKit server URL, API key, and API secret, from your LiveKit Cloud project settings
- Deepgram, ElevenLabs, and Google API keys, from your account with each provider
- ElevenLabs voice ID, optional. Leave it blank unless the agent connects but stays silent
- Anam API key, optional. Leave it blank to run audio only, with no avatar
- Pega server hostname, OAuth client ID and secret, and application alias, from your Pega OAuth 2.0 client using the client-credentials grant
- Number of agent instances. One is enough to start

Select Create Stack. The template provisions its own VPC, so there is nothing else to prepare. Watch the stack status until it reads CREATE_COMPLETE, either in the CloudFormation console or with this command.

```
aws cloudformation describe-stacks --stack-name <your-stack-name> \
  --query 'Stacks[0].StackStatus' --output text
```

Step 3: Find Your Running Task

List the running tasks in the cluster the stack created. The cluster name is the stack's ClusterName output.

```
aws ecs list-tasks --cluster <your-stack-name>-cluster --desired-status RUNNING
```

Copy the task ID from the returned ARN, then confirm both containers are healthy.

```
aws ecs describe-tasks --cluster <your-stack-name>-cluster --tasks <task-id> \
  --query 'tasks[0].containers[*].[name,lastStatus,healthStatus]' --output table
```

- cara-workflow-gateway should report HEALTHY
- cara-agent-worker reports UNKNOWN. That is expected, because it defines no ECS health check

Step 4: Port-Forward to the Gateway

The gateway exposes no public port by design. It is outbound only, with no inbound ports, so you reach it through ECS Exec and SSM port forwarding. Start by getting the gateway container's runtime ID.

```
aws ecs describe-tasks --cluster <your-stack-name>-cluster --tasks <task-id> \
  --query "tasks[0].containers[?name=='cara-workflow-gateway'].runtimeId" --output text
```

Open the tunnel, and leave it running in its own terminal.

```
aws ssm start-session \
  --target "ecs:<your-stack-name>-cluster_<task-id>_<runtime-id>" \
  --document-name AWS-StartPortForwardingSession \
  --parameters '{"portNumber":["8080"],"localPortNumber":["8080"]}'
```

The session reports that it is waiting for connections. Confirm the tunnel works.

```
curl http://localhost:8080/healthz
```

In production, your backend server calls the gateway directly inside AWS. There is no port forwarding involved. Both your application and the Cara stack you create must run in the same AWS region, so your backend reaches the gateway over the internal network. The simplest approach is to place your backend in the same VPC that the stack created (or peer your existing VPC with it), then call the gateway using its ECS service discovery address or private IP on port 8080.

Tunnel Troubleshooting

If the tunnel does not open, or curl hangs or is refused, check whether anything is already bound to the port. A tunnel left running from an earlier attempt is the usual cause.

```
lsof -i :8080
```

If a process is holding the port, stop it and start a fresh tunnel.

```
pkill -f "session-manager-plugin"; pkill -f "aws ssm start-session"
```

If a task was replaced by a stack update or a restart, its runtime ID changes. Run the Step 4 commands again against the task that is currently running, not the earlier one. To rule out the tunnel and test connectivity from inside the container, use ECS Exec.

```
aws ecs execute-command \
  --cluster <your-stack-name>-cluster --task <task-id> \
  --container cara-workflow-gateway --command "/bin/sh" --interactive
```

Step 5: Confirm Pega Is Reachable

Retrieve the agent key that CloudFormation generated, then call the gateway's Pega case types endpoint. A successful call returns your Pega case types.

- A 504, or a timeout, usually means the network security policy on your Pega instance does not allow inbound traffic from this task's public IP. Check the allowlist with whoever manages that instance
- A 502 wrapping a 401 usually means the client-credentials grant is not enabled on your Pega OAuth client

Step 6: Test a Call

Open the hosted test console page. The page itself walks you through the final two steps.

- Copy the curl command shown, using the TestTokenKey value you set as a stack parameter to enable this test path
- Run the command in your terminal, with the tunnel from Step 4 still open, then paste the returned url and token into the page

- Select Start Test Call, allow microphone access, and speak. The agent should respond within a few seconds
- If the call connects but the agent stays silent, the default ElevenLabs voice may not be available to your account. Free-tier ElevenLabs keys work only with voices you own, not with the library default. Set the stack's TtsVoiceId parameter to a voice ID from your own ElevenLabs account, then redeploy

You Are Done

Worker to LiveKit, LiveKit to your browser, and gateway to Pega are all confirmed working. To connect real traffic, see Connecting Your Channels in DEPLOYMENT.md, which covers phone service over SIP and your own web frontend.