CodeGen Agent – AgentCore POC Setup & Run Guide

1. Project Structure

```
Projects/
└── AI/
    ├── code_gen_agent.py
    ├── venv/          (Python env for development)
    └── vagent/        (Python env for AgentCore runtime)
```

---

2. Activate Environments

Activate development environment:

```
source venv/Scripts/activate
```

Activate AgentCore environment:
``

Here is the **entire README as one single, clean, correctly formatted file**.
This is **copy-paste ready** for GitHub — numbering works, spacing works, and formatting is polished.

CodeGen Agent – AgentCore POC Setup & Run Guide

1. Project Structure

```
Projects/
└── AI/
    ├── code_gen_agent.py
    ├── venv/          (Python env for development)
    └── vagent/        (Python env for AgentCore runtime)
```

---

2. Activate Environments

Activate development environment:

```

```

```
source venv/Scripts/activate
```

Activate AgentCore environment:

```
source vagent/Scripts/activate
```

---

3. Configure & Launch the CodeGen Agent

Configure the agent:

```
agentcore configure -e code_gen_agent.py
```

Launch the runtime:

```
agentcore launch
```

---

4. Why We Switched from S3 Direct Code Deploy to Docker Container Deploy

Initially we attempted **Direct Code Deploy** (S3-based deployment).
However, this approach failed due to **package size limitations**.

Root Cause

The CodeGen Agent uses several heavy Python dependencies, including:

- boto3
- bedrock_agentcore
- strands
- fastapi / pydantic
- supporting utilities

These made the deployment bundle exceed the size allowed for S3 direct deployment.

AgentCore Behavior

AgentCore automatically detected that the code bundle was too large and **fell back to container-based deployment**.

Why Docker Deployment Works

Docker containers overcome all the limitations because they:

- support large dependency sets
- can include native binaries
- can ship compiled wheels
- allow any project size
- provide a clean, reproducible environment

Summary

S3 Direct Deploy fails due to size limits.

Docker deploy succeeds because it supports larger, dependency-heavy codebases.

---

5. Error Fixes During Setup

Error 1 — Missing `uv`

```
Warning: Direct Code Deploy deployment unavailable (uv not found).
```

Fix:

```
pip install uv
```

Error 2 — Missing `zip` Utility

```
Warning: Direct Code Deploy deployment unavailable (zip utility not found).
```

Install GnuWin32 Zip:

```
winget install GnuWin32.Zip
```

Add to PATH:

```
export PATH="$PATH:/c/PROGRA~2/GnuWin32/bin"
```

---

6. Test Payload

Use this to test the agent:

```
{
  "requirement": "Create a simple FastAPI service with one GET /health endpoint."
  "language": "python"
}
```

---

7. Runtime Initialization Timeout Error

```
Unable to invoke endpoint successfully:
Runtime initialization time exceeded.
```

This means the CodeGen Agent didn't complete startup within the 60-second initialization window.

Common Causes

- Heavy imports at module load
- Large number of dependencies
- Slow initialization logic inside global scope
- Missing or misplaced entrypoint (`if __name__ == "__main__": app.run()`)
- Unoptimized base image in the container

If you want, share your latest `code_gen_agent.py` and I'll help optimize startup time.