Remote Development in Container Runtime

Container Runtime Remote Dev is a remote execution service that runs inside Snowpark Container Services executing a Container Runtime image. This allows you to develop locally (using an IDE such as VSCode or Cursor) and execute directly in the Container Runtime ecosystem with full access to your data, compute resources, and a persistent development environment, leveraging the full power of distributed ML, flexibility in packages and compute, and many other optimizations. If you prefer a web-based IDE, you can access a full web-based VS Code through your browser, entirely served from within a Snowflake service

Keep using your familiar local VS Code or Cursor, but execute code in Snowflake's cloud environment with full access to data and compute.

Benefits

- **Local IDE Integration**: Connect your local VS Code/Cursor to a Container Runtime instance (via SSH) and continue using your familiar development environment
- **Stage-based Persistent Storage**: Your workspace and VS Code settings persist across sessions using Snowflake stages
- **Flexible Compute Options**: Choose from various CPU and GPU compute pools based on your needs
- **Secure**: Configure external integrations for secure internet connectivity and use your existing Snowflake credentials

Prerequisites

⚠️ **Important**: Remote dev is currently in private preview. Setup and APIs may change.

1. SnowCLI with Remote Dev Plugin
    - ***Note*****:** To enable the Remote Dev environment, a custom SnowCLI branch is required instead of the released version. Please follow these steps even if SnowCLI is already installed.
    - Go to Snowflake CLI repository
    - Switch to the `remote_dev_prpr` branch

```shell
# 1. Clone and switch to customized branch
git clone https://github.com/snowflakedb/snowflake-cli
cd snowflake-cli
git checkout remote_dev_prpr

# (Optional) If you already have the existing local branch before and want to update it
git fetch origin
git reset --hard origin/remote_dev_prpr

# 2. Install dependencies
pip install -U hatch
```

```
# 3. Choose one:
# (Option 1: Create a transient virtual environment with installation)
hatch shell
# (Option 2: Install the snowcli into your python environment)
hatch build && pip install .
```

2. If you have an existing Snowflake Connection for SnowCLI skip this step. If not, make sure to configure one:

```
Shell
       snow connection add
```

Once you follow the prompts and finish the setup, your `~/.snowflake/config.toml` should looks like:

```
None
default_connection_name = "my_connection"

[connections.my_connection]
host = "<YOUR_HOST>"
account = "<YOUR_ACCOUNT>"
user = "<YOUR_USER>"
password = "<YOUR_PASSWORD>"
warehouse = "<YOUR_WAREHOUSE>"
role = "<YOUR_ROLE>"
database = "<YOUR_DATABASE"
schema = "<YOUR_SCHEMA>"
```

3. Install websocat (for local VS Code connection)

```
None
# macOS
brew install websocat

# Other platforms: download from https://github.com/vi/websocat/releases
```

4. Set up an External Access Integration and configure a Compute Pool for your remote execution service. Note that EAIs are required if you want your remote session to `pip install` from external package repositories.

Executing a Remote Session on VS Code

Step 1: Create Remote Environment

```
None
snow remote start \
  my_remote_dev \
  --compute-pool YOUR_COMPUTE_POOL \
  --eai-name ALLOW_ALL_INTEGRATION \
  --stage YOUR_SSE_STAGE \
  --ssh
```

What this does:

- Creates a service named `my_remote_dev` that runs a Container Runtime instance (along with a Ray cluster) over the specified compute pool (on a single node)
- Mount a server-side-encrypted stage to the service workspace
- Enables an SSH connection

💡 **Tip**: Use EAI `ALLOW_ALL_INTEGRATION` to download VS Code extensions in the remote environment. This is **required** for remote SSH setup.

Example:

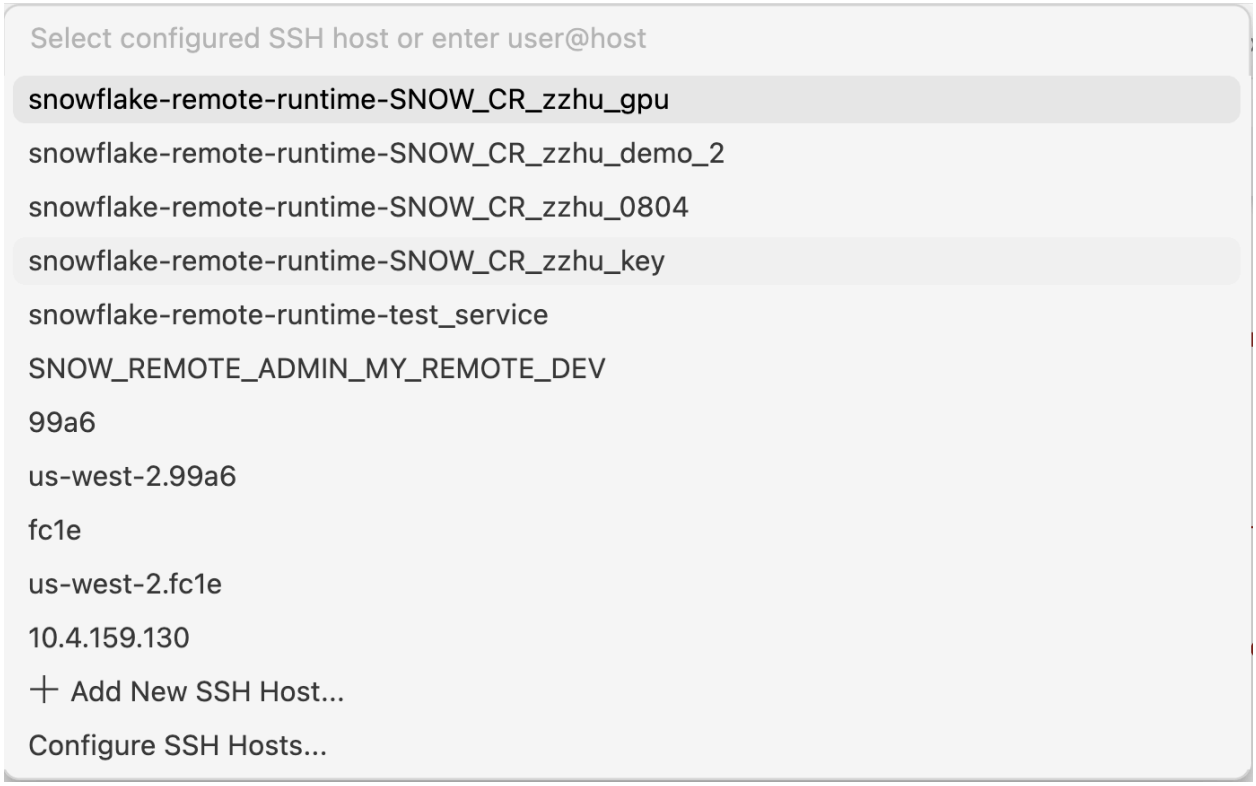
```
(snowflake-cli) (base) → snowflake-cli git:(08-17-enable_remote_ssh_connection) snow remote start \
  my_remote_dev \
  --compute-pool E2E_CPU_POOL \
  --stage @zzhu_container/test \
  --eai-name ALLOW_ALL_INTEGRATION \
  --ssh

Creating remote development environment 'SNOW_REMOTE_ADMIN_MY_REMOTE_DEV'...
Waiting for service SNOW_REMOTE_ADMIN_MY_REMOTE_DEV to be ready...
✓ Remote Development Environment SNOW_REMOTE_ADMIN_MY_REMOTE_DEV created successfully!
VS Code Server URL: jyi4q3j-sfengineering-notebook-mltest.awsuswest2preprod8.pp-snowflakecomputing.app
Starting SSH session management for 'SNOW_REMOTE_ADMIN_MY_REMOTE_DEV'...
You can now connect using: ssh SNOW_REMOTE_ADMIN_MY_REMOTE_DEV
Press Ctrl+C to stop SSH session management
SSH configuration updated (refresh #1)
```

Step 2: Connect from Local VS Code

1. Open VS Code
2. Install the Remote-SSH extension or the Remote Development extension pack.

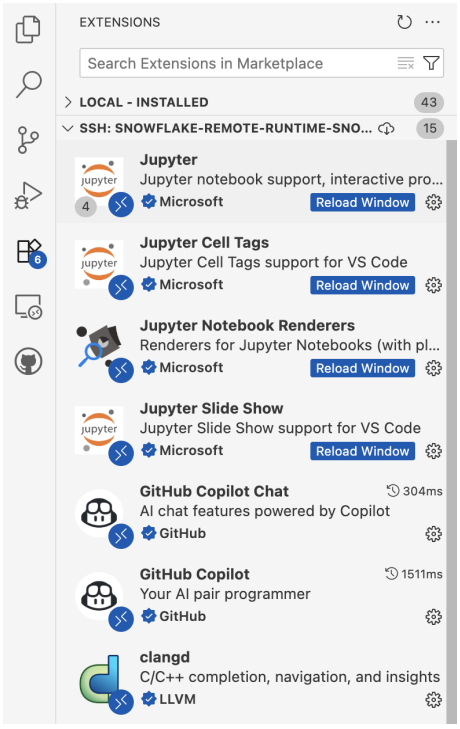
3. **Cmd/Ctrl + Shift + P** → "Remote-SSH: Connect to Host"



4. Select `SNOW_REMOTE_ADMIN_MY_REMOTE_DEV`
5. You're now coding locally but executing remotely!

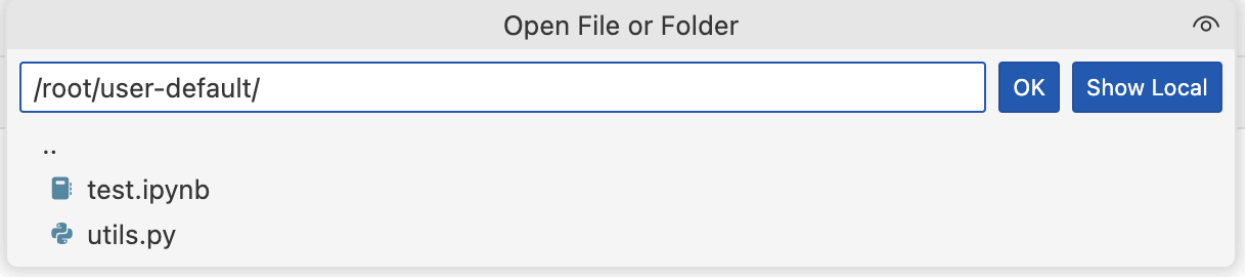
Step 3: View Extensions

Once connected, install extensions as you normally would - they'll run in the remote environment with access to your remote files.



Step 4: Open folder and start coding

Open the `/root/user-default/` folder. All the files in it will be persisted in the given stage.



💡 **Tip**: If python kernel is not automatically detected for .ipynb file, it might due to the python extension needs update. Check your extension list and update it if available.

Step 5: Monitor Remote Services

```
None
-- Check your running services
snow remote list
```