Leveraging GenAI Tools on FASRC Clusters

Manasvita Joshi & Nathan Weeks

Harvard - FAS Research Computing

Learning objectives

- AI Terminology
- Harvard's GenAI Guidance
- OpenAI API Key - CLI
- AWS Bedrock
  - CLI
  - OOD - Jupyter-AI Extension
- RStudio Server
- VSCode
  - Codex, Claude Code
- Ollama & dev containers

Training Material

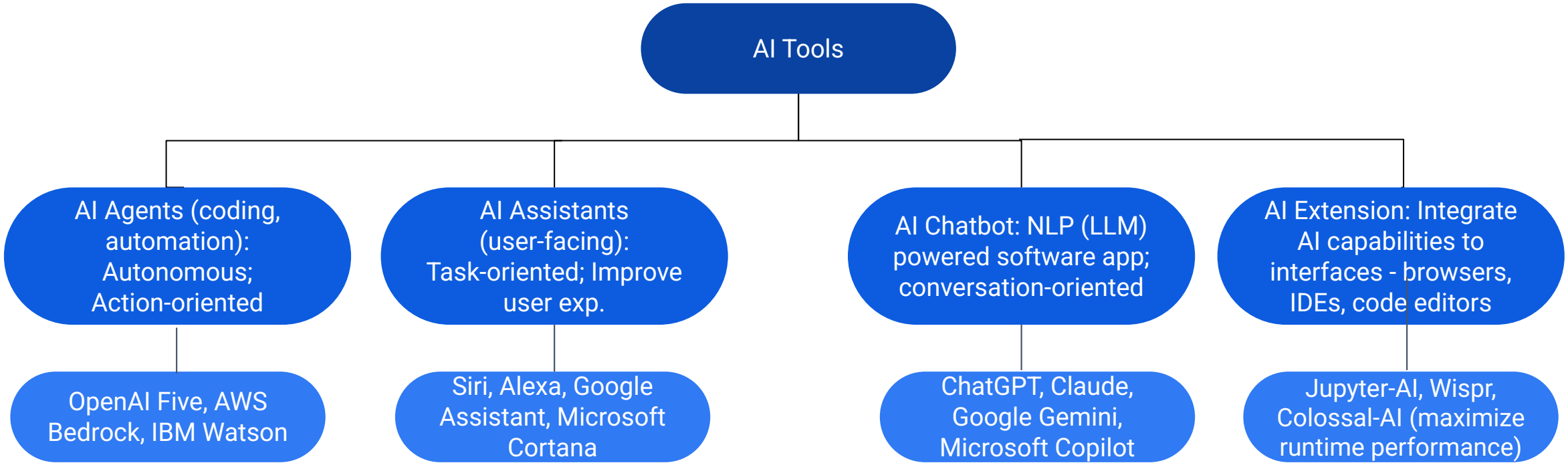
```
# Login to Cannon
ssh <username>@login.rc.fas.harvard.edu

# Check current location; change if desired:
> pwd
> cd <desired-location>

# Clone FASRC User_Codes repository: https://github.com/fasrc/User_Codes
HTTPS - git clone https://github.com/fasrc/User_Codes.git

# In <desired-location>, go to GenAI-Tools folder in User_Codes:
> cd User_Codes/Training/GenAI-Tools
```

- Training Materials – FASRC DOCS - Slide Deck

AI Tool vs AI Model: Car & Engine

- AI Model (The “Engine”)
  - Algorithms/computer programs trained on large datasets to predict or generate content
  - Learn patterns, analyzes inputs, produces meaningful outputs
  - Not visible to the end user but powers the AI tool from “under the hood”
  - GPT, Llama, Claude, BERT, etc,
- AI Tool (The “Car”)
  - User-facing application that utilizes the model to solve the problem initiated by the user
  - Provides an interface to the users to interact with the underlying model
  - ChatGPT, Notion AI, Jasper, etc.
- Use AI tool to interact with AI model to get work done
- Checkout:
  - https://medium.com/@akinbondabefe/ai-tools-vs-ai-models-think-car-engine-ef0071eabd77
  - https://www.linkedin.com/pulse/ai-tools-vs-models-whos-driving-future-work-arnab-bhor-8cpec/

AI Tools



- Finer Distinction: Table in [Understanding the differences between AI agents, assistants, and other intelligent tools - Anima Blog]

Harvard's Generative AI Guidelines

- Protected data (Level 2 & above) not for publicly available GenAI tools
- Review content before propagation
- Review your division/school/lab policy on the use of GenAI tools
- Lookout for phishing
- **Use approved tools & registered API keys for Harvard work**

Register an API key with HUIT

- Secure access; Proper billing; Grant compliance

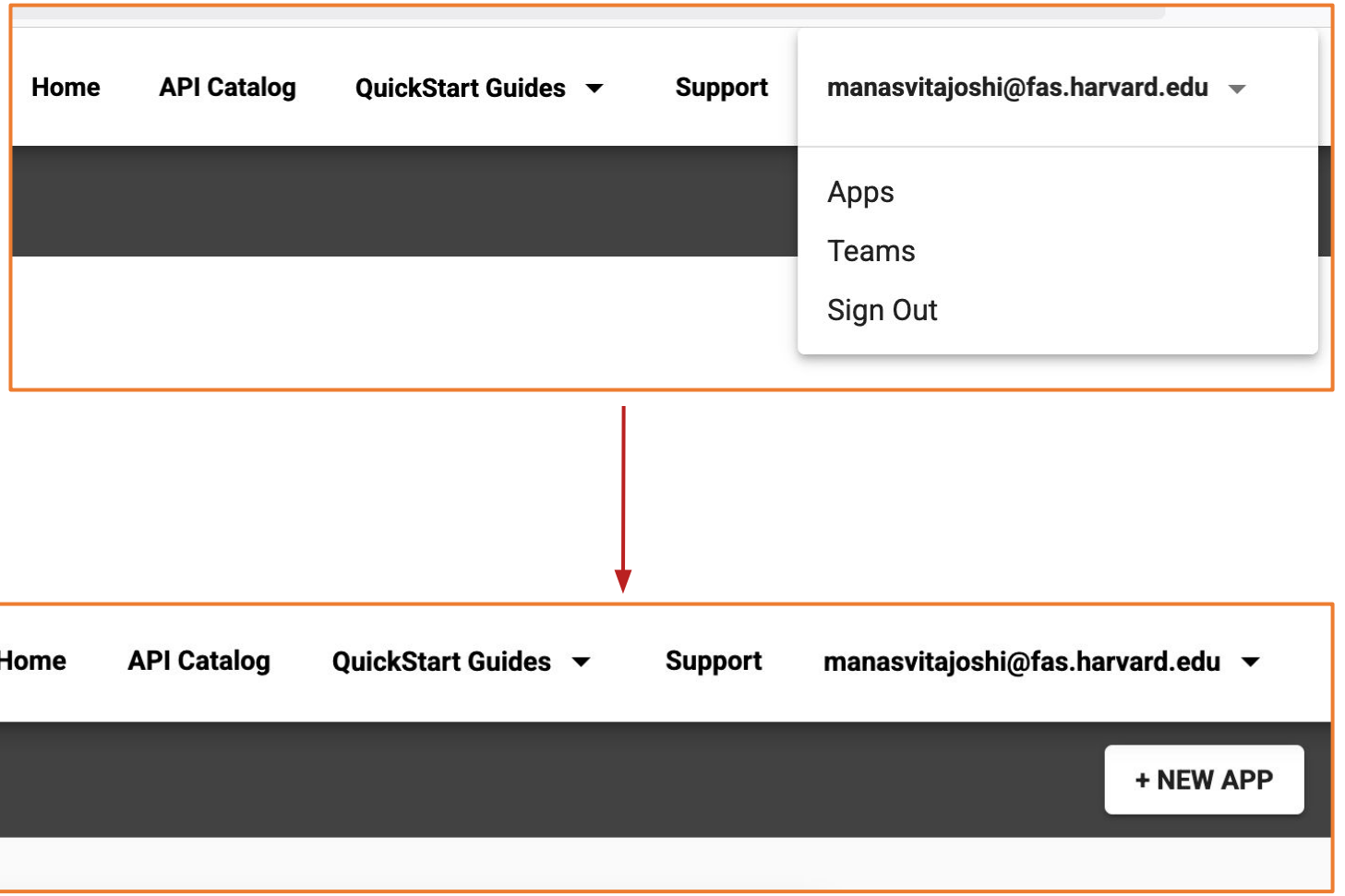
- HUIT AI Services: API Portal
  - Direct API - OpenAI
  - Indirect API - via AWS Bedrock

- OpenAI:
  - FASRC Guidelines for OpenAI Key and Harvard Agreement
  - Option 2 - Individual (PI/lab account) or Team account (developer teams)

- AWS Bedrock:
  - Similar to above: Setting Up Anthropic API Access for Harvard Users — Generative AI for Scholarship
  - Follow Step #1 & #2 (#3 is handled in scripts)

OpenAI for Community Developers API key

- For FAS users - up to $10 credit per month
- https://portal.apis.huit.harvard.edu/docs/ais-openai-direct-limited/1/overview - Harvard Key sign in required



Direct API - OpenAI (CLI)

- **Install OpenAI Python API Library**: OpenAI – FASRC DOCS

```
# Login to Cannon
ssh <username>@login.rc.fas.harvard.edu

# Go to a compute node on the test partition:
salloc --partition test --nodes=1 --cpus-per-task=2 --mem=8GB --time=01:00:00

# Load the latest/ default Python module & disable ~/.local from being used:
module load python
export PYTHONNOUSERSITE=yes

# Create a mamba environment & install openai python API library:
mamba create -n openai_env openai -y
```

Direct API - OpenAI (CLI)

- **Run OpenAI** - Use Harvard registered API key
- API key generated on [OpenAI Platform] is **not** registered under Harvard agreement
- OpenAI example: https://github.com/fasrc/User_Codes/tree/master/AI/OpenAI
- Unable to create env?

```
mamba activate
/n/netscratch/rc_admin/Everyone/genai-training/openai_env
```

```
# Request an interactive job
salloc --partition test --time 01:00:00
--mem=8GB --cpus-per-task=2

# Activate mamba environment
mamba activate openai_env

# Replace my_key with your lab's registered key
export OPENAI_API_KEY='my_key'

# Set SSL_CERT_FILE with system's certificate
export
SSL_CERT_FILE='/etc/pki/tls/certs/ca-bundle.crt'

# Run OpenAI example
python openai-test.py
```

Indirect - AWS Bedrock (CLI)

- **Install Anthropic Python SDK**: Anthropic – FASRC DOCS

```
# Login to Cannon
ssh <username>@login.rc.fas.harvard.edu

# Go to a compute node on the test partition:
salloc --partition test --nodes=1 --cpus-per-task=2 --mem=8GB --time=01:00:00

# Load the Python module & disable ~/.local from being used:
module load python
export PYTHONNOUSERSITE=yes

# Create a vanilla mamba environment with latest Python version:
mamba create -n claude_env python -y
```

CLI - AWS Bedrock (CLI)

- **Run Anthropic** - Use Harvard registered API key
- API key generated on Anthropic API Platform is **not** registered under Harvard agreement
- Anthropic example: https://github.com/fasrc/User_Codes/blob/master/Training/GenAI-Tools/anthropic-bedrock-example.py
- Unable to create env:

```
mamba activate
/n/netscratch/rc_admin/Everyone/genai-training/claude_env
```

```
# Request an interactive job
salloc --partition test --time 01:00:00
--mem=8GB --cpus-per-task=2

# Activate env & install Anthropic API Python
library
mamba activate claude_env
pip install anthropic

# Set SSL_CERT_FILE with system's certificate
export
SSL_CERT_FILE='/etc/pki/tls/certs/ca-bundle.crt'

# Run Anthropic example
python anthropic-bedrock-example.py
```

Jupyter-AI Extension

- Accelerate full-stack software development, pair-programming, & analysis
- Explore GenAI models in your application
- Open source tool to integrate GenAI models into Jupyter's IDEs - lab & notebook

```
# Go to compute node on test partition:
salloc --partition test --nodes=1 --cpus-per-task=2
--mem=8GB --time=02:00:00

# Load Python module & disable ~/.local from being
used:
module load python
export PYTHONNOUSERSITE=yes

# Create mamba env in your lab folder:
mamba create -y -p
/n/holylabs/<PI_lab>/Lab/jupyter-ai-liteenv
python=3.13

# Activate env & install packages:
mamba activate
/n/holylabs/<PI_lab>/Lab/jupyter-ai-liteenv

pip install ipykernel 'jupyter-ai==3.0.0b9'
'jupyter-ai-magics==3.0.0b7' boto3
```

Jupyter-AI Extension using OOD

- Accessing OOD from Cannon
  - Connect to FASRC VPN - Virtual Desktop (VDI) through Open OnDemand – FASRC DOCS
  - Then go to https://rcood.rc.fas.harvard.edu
- Accessing OOD from FASSE
  - Connect to FASSE VPN - FASSE VDI Apps – FASRC DOCS
  - Then go to https://fasseood.rc.fas.harvard.edu
- Set FASSE proxy: FASSE Proxy Settings – FASRC DOCS

Filling a form to launch an app

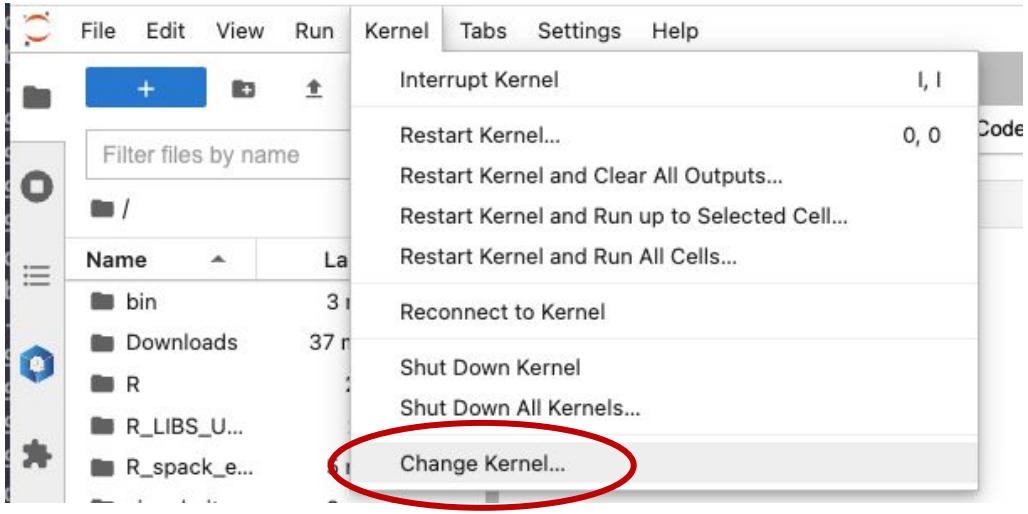
- Request the resources that you need
  (If you don't know for a first trial run, use similar resources as your laptop/desktop)
  - Partition (Name): depends on Cannon (URL) vs FASSE (URL)
  - Memory (RAM): amount of memory in GB
  - Number of cores: recommended at least 2
  - Number of GPUs: if >= 1, make sure you **select** a gpu partition
  - Allocated time: time you would like your session to run

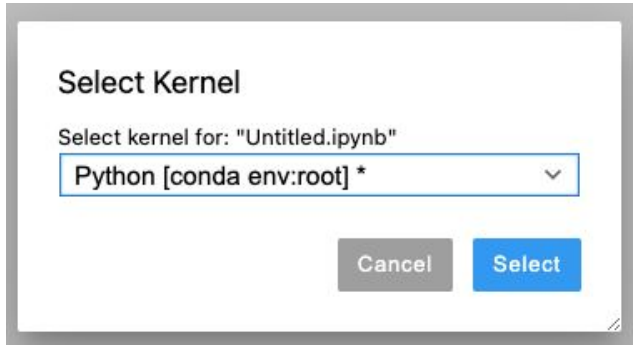
  } the minimum and/or maximum values of each field depends on the selected partition

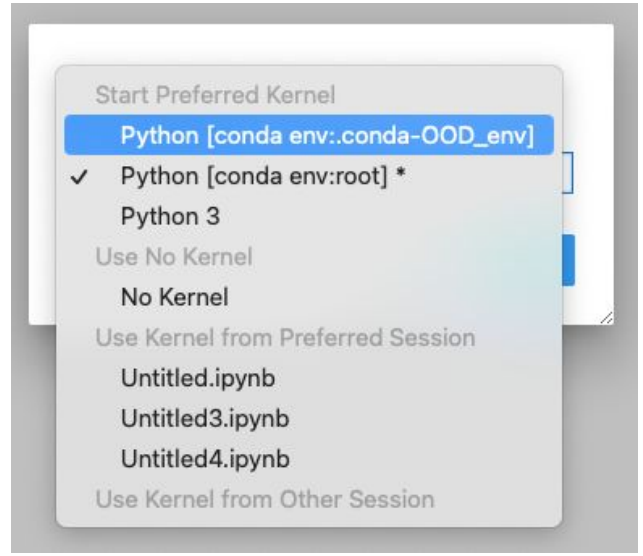
  - Email for status notification: to know when job starts, ends
  - Reservation: if you have a special reservation (this requires approval from FASRC)
  - Account: use this if you have more than one PI_lab affiliation

Jupyter Notebook

- Launch **new** Jupyter Notebook session (existing session will not work!)
- Select newly created conda environment as the kernel
  a. Open a notebook
  b. On the top menu, click Kernel -> Select Kernel -> Click on OOD_env
  c. Note: kernels is the same as conda, python, mamba environment







Example Demo

- Load & time the loading of jupyter_ai_magics_command package
- Part 1: OpenAI using Community Developers HUIT API key
- Part 2: Anthropic via AWS Bedrock HUIT API key
- Part 3: Accessing Bedrock providers without Jupyter-AI extension

Closing running OOD windows/tabs

- In most OOD apps, you can close the browser tab while the code is running, and the code will continue to run on the background
- Jupyter Notebook will not! The cell that is running will lose the data and output files will not be written
  - Solution: run Remote Desktop app and launch Jupyter Notebook from within Remote Desktop
  - Open OnDemand (OOD/VDI) Remote Desktop: How to open software – FASRC DOCS

RStudio Server & GenAI

- GitHub Copilot
  - Integrated into RStudio Server IDE; supports:
    - Code suggestions
    - Questions (simple question/answer prompts)
  - For Harvard work, requires GitHub Copilot for Business license from HUIT
- chattr
  - R package for interacting with LLMs from various AI platforms
    - For Harvard work, use Harvard API portal to obtain API key
  - Maintained by posit
  - Supports exploratory data analysis:
    chattr enriches your request with... name and structure of data frames currently in your environment, the path for the data files in your working directory.

Claude Code for VS Code

Update settings **in the remote**:

1. check `Claude Code: Disable Login Prompt`
2. Under Claude Code: Environment Variables, click "Edit in settings.json", and fill in:

```
"claudeCode.environmentVariables": [
    {"name": "ANTHROPIC_BEDROCK_BASE_URL",
     "value": "https://apis.huit.harvard.edu/ais-bedrock-llm/v2"},
    {"name": "ANTHROPIC_API_KEY",
     "value": "...Your API Key..."},
    {"name": "ANTHROPIC_SMALL_FAST_MODEL",
     "value": "us.anthropic.claude-opus-4-5-20251101-v1:0"},
    {"name": "CLAUDE_CODE_SKIP_BEDROCK_AUTH",
     "value": "1"},
    {"name": "CLAUDE_CODE_USE_BEDROCK",
     "value": "1"}
]
```

Codex

- Codex CLI
  - ChatGPT EDU
  - HUIT AI Services (Bedrock or OpenAI) through Harvard API portal

- Codex VS Code extension
  - ChatGPU EDU
  - HUIT AI Services - “OpenAI Direct” only (requires *streaming*) (untested)

Cluster Best Practices

- Use HUIT configured API key to access GenAI tools on FASRC clusters
  - Ensures working with DSL2+ data isn't a problem
  - **Default Rule**: "Assume what you type into a prompt becomes public training data unless verified otherwise."
- DO NOT hardcode API keys in scripts or Jupyter Notebooks
  - Use export commands in your .bashrc or use environment variables to store keys securely or have the script prompt you to set up the keys (restricted to interactive)
- DO NOT share keys in notebooks or emails
- DO NOT upload .env files to GitHub or shared Lab folders
  - Commit keys to GitHub/GitLab

Best Practices Contd.

- While using Claude, be wary of `--dangerously-skip-permissions`
  - Could delete files/folders
  - Could introduce malware on the cluster if agent(s) not set up to check
- Best to use GenAI tools & agents within a sandbox or a container
  - VSCode Remote Development via SSH and Tunnel – FASRC DOCS
  - When possible, use: Sandboxing - Claude Code Docs, “Default Permissions” (Codex)
- AI coding assistants are memory-intensive
  - Compute node should be used instead of the login node
  - Command line access with Terminal (login nodes) – FASRC DOCS
  - Current limit set to 4GB in memory & 5 sessions per user
  - Use interactive (CLI, OOD) or batch scripts to request **only** the resources you need

Best Practices Contd.

- Idle VSCode/Jupyter sessions with loaded models waste GPU cycles & block others from using a shared resource, specifically for local LLM
  - LLMs occupy space & so do your models. Be aware of VRAM & cluster storage limits
- Environment Isolation: Use Mamba or other virtual envs to avoid dependency hell
- Cost Monitoring: If using APIs, be aware of your lab's limits
  - For example, Claude's `/cost` command. See https://astrostubbs.github.io/GenAI-for-Scholarship/session3-power-user.html

Save on cost - optimize on prompts

- Prompt Engineering
  - # ❌ Vague prompts - "Fix my code"
  - # ✅ Clear, specific prompts

```
"""
I have a pandas DataFrame with columns ['date', 'value',
'category'].
I need to:
1. Filter for category 'A' only
2. Calculate 7-day rolling mean
3. Handle NaN values
Current error: TypeError on groupby
Please provide corrected code.
"""
```

- Specific details over keywords (it’s not a Google search!)

Best Practice Contd.

- ✅ Use for brainstorming and code generation
- ✅ Review all generated code carefully & validate results
  - Test on sample data
  - Check for edge cases/boundary conditions
- ✅ Rotate keys regularly
- ✅ Use organization/team API keys
- ✅ Monitor API usage for unauthorized access
- ❌ Don't blindly trust outputs
- ❌ Don't paste sensitive data into prompts

Data Privacy

- Protecting Research Data - Understand What Leaves Your Cluster
- Use Harvard-approved and managed GenAI tools for all Harvard work
- Do not enter Data Security Level (DSL) 2–3 data into public GenAI services or personal accounts
- Do not enter DSL 4+ data into any GenAI service (including Harvard-managed tools)
- Consult your IRB for research with human subjects; PHI/PII requirements
- Check lab security policies & funder requirements (NIH, NSF, etc.)
- Never include proprietary research data or unpublished results

Ethical Considerations

- Academic Integrity:
  - ✅ Use GenAI for assistance and learning
  - ✅ Disclose AI usage in methods/acknowledgments
  - ❌ Don't submit AI output as your own work
  - ❌ Don't use to bypass learning objectives

- Research Integrity:
  - GenAI should enhance, not replace, critical thinking
  - Always validate outputs
  - Document your process
  - Be transparent with collaborators

Ethical Considerations

- Environmental Considerations:
  - Local models reduce cloud energy usage
  - Batch API calls efficiently
  - Use appropriate model sizes
  - Consider computational footprint
- Attribution:
  - Portion of this slide deck generated using Anthropic claude-haiku-4-5 via Jupyter-AI extension. All content reviewed & validated by presenters

FASRC documentation

- FASRC docs: FASRC DOCS
- Training
  - Calendar: Training Calendar | FAS Research Computing
  - Material: Training Materials – FASRC DOCS
- Getting help
  - Office hours: Virtual Office Hours | FAS Research Computing
  - Ticket:
    - HUIT Service Portal -> Submit Ticket: Submit Ticket - IT Help
    - Email: rchelp@rc.fas.harvard.edu

Training session evaluation

Please, fill out our training session evaluation. Your feedback is essential for us to improve our trainings!!

https://tinyurl.com/FASRC-training

Thank You!

Questions? Comments?

Supplemental Content

Codex CLI using HUIT AI Service endpoint

1. Install codex CLI

```
module load python
mamba create -n codex codex
mamba activate codex
```

Codex CLI using HUIT AI Service endpoint

2. Generate API key (for an OpenAI service) in Harvard API Portal:
https://portal.apis.huit.harvard.edu/

3. Add to ~/.profile (or ~/.bash_profile)

```
export HARVARD_OPENAI_API_KEY=...API Key from Harvard API Portal…
```

4. Source ~/.profile (~/.bash_profile) (or start a new login shell)

```
source ~/.profile
```

Codex CLI using HUIT AI Service endpoint

5. Add to ~/.codex/config.toml (`mkdir ~/.codex` if the directory doesn't exist)

```
model_provider = "harvard_openai"
model = "gpt-5.3-codex"

[model_providers.harvard_openai]
name = "Harvard OpenAI Gateway"
base_url =
"https://go.apis.huit.harvard.edu/ais-openai-direct-limited-schools/v1"
wire_api = "responses"
env_http_headers = { "api-key" = "HARVARD_OPENAI_API_KEY" }
```

Codex CLI using HUIT AI Service endpoint

4. Run codex command:

```
codex
```

Ollama - Open Source LLMs

- AI tool to download, run, & manage open source LLMs locally
- Why run local?
  - **No API costs, total data privacy**

- On the cluster, need to be run via Singularity on a compute node
  - Pull Docker container into Singularity
  - Spin up the Ollama server in one terminal
  - Open another terminal and ssh to the same host
  - Start a Singualrity shell
  - Run ollama on open source LLM of your choice

- https://github.com/fasrc/User_Codes/blob/master/AI/AITools/ollama.md

- Another tool: vLLM

Dev Containers

- VS Code Dev Containers extension "lets you use a Docker container as a full-featured development environment"
- A way to sandbox AI coding assistants / agents
  - Only repository with devcontainer.json is accessible by default

See Using Dev Containers on the FASRC Cluster

Example:
User_Codes/Training/GenAI-Tools/devcontainer.json

After configuring VS Code settings:

```
sbatch devcontainer.job
```