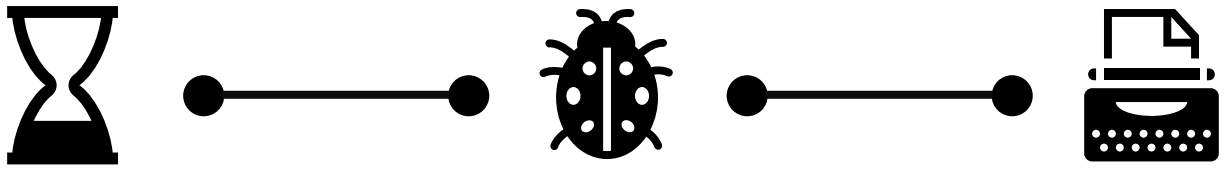
NAMS-2
NASA Academic Mission Services Contract

Claude Code: Introduction and Demonstration

Mohamad Refai
Crown Innovations, Inc.

CROWN
AIR MOBILITY. ADVANCED.

AI Assisted Development



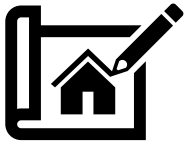
Faster Development

Earlier Detection

Automated Documentation

Ai Assisted Development for Airspace Tools: What it does, how we use it, what we've learned, & a demonstration

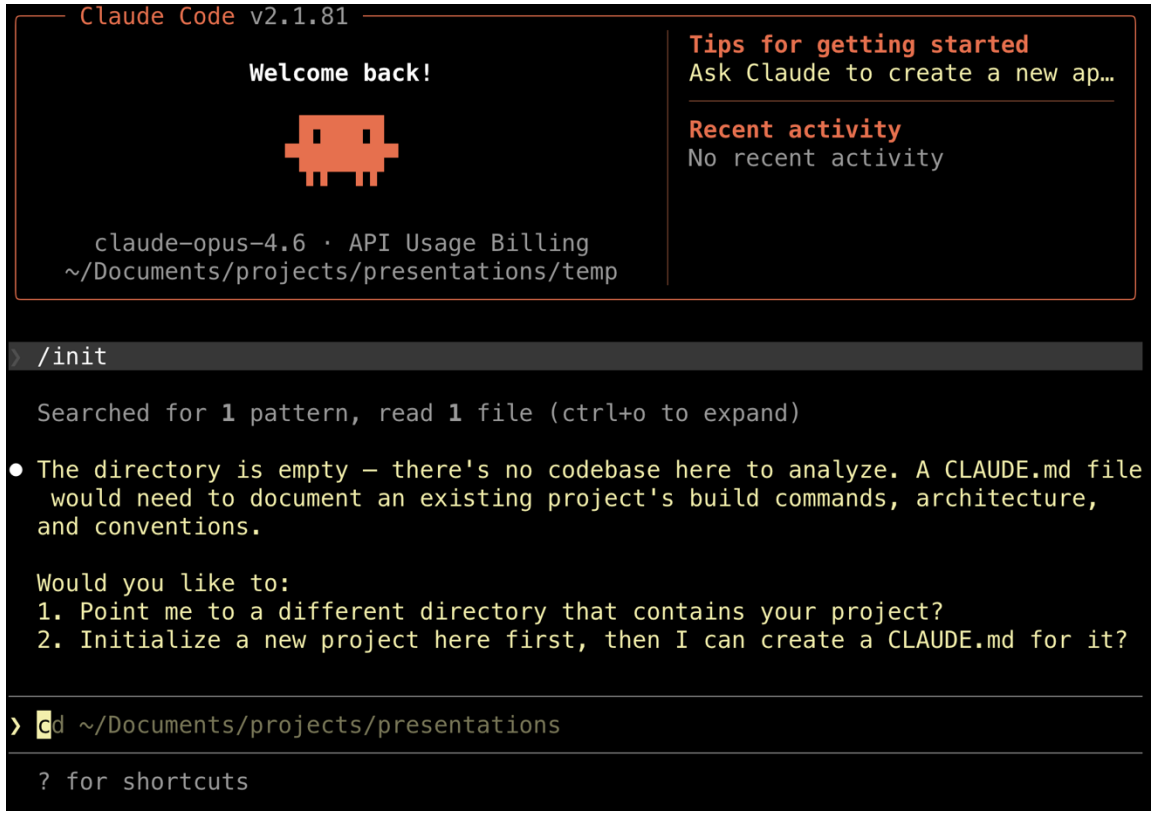
How the SHIFT Team is Using Claude Code

Plan	Design	Build	Test	Document	Deploy
					
Task Plans	Service Design Analysis	Coding Refactoring Analysis	Unit Testing Code Review V&V	Code Comments Design Docs Plan Docs	(future)

The ATMS SHIFT team has been using Claude in all aspects of the SDLC

ATMS: Air Traffic Management and Safety
SHIFT: Strategic Harmonization for Integrated Flow and Trajectories

What is Claude Code?



- It is AI-powered
- It is an Agentic coding assistant
- It can act independently
- It can make decisions independently
- It can run tasks to completion w/o supervision
- It runs in a terminal window

Plugins are available for integration into IDEs (eclipse, VSCode, JetBrains)

Claude Code Use Cases

NAMS-2
NASA Academic Mission Services Contract

Research

- Document Processing
- Complex Reasoning
- Machine Learning & Data Science
  - Scaffolding & Automated Workflows

Software Engineering (SWE)

- Complex Refactoring
- Bug Detection & Bug Fixing
- Adversarial Testing
- Design and Scaffolding
- Project Planning & Documentation

Information Science

- Information Management
- Knowledge Engineering

Workflow & Productivity

- Multi-tasking
- Training Material Preparation

Claude Code: Benefits

Developer must provide context, review, and iterate

Task	Unassisted	AI Assisted (Claude Code)
Coding	5 days	4.5 days
Documentation	5 pages	6 pages
Test Coverage	Manual	Automated (higher coverage)
Bug Discovery	Late	Early
Code Review	Peer	AI + Peer
One-off Tools	Not cost effective	Fast

AI actions are fast, but developer review and iteration times increase

Productivity will improve as prompts and context precision improve

The Claude Experience

Installation
& Setup

Code Style
Prompts
Iteration

Customization
Skills, MCP
Sub-agents

Automation
Hooks, Loops, etc.
Assisted Personalization

* MCP: Model Context Protocol

Claude Models

	Opus	Sonnet	Haiku
	 Slower	 Moderate	Fast
	 Smartest	 Capable	 Routine
	**$$$**	**$$**	**$**
	Deep Reasoning Design Decisions Complex Refactor Autonomous Agents	Coding Debugging Developing Features Data Analysis	Worker Bee Simple Tasks Editing Module Scaffolding

Claude Code Basics

Tokens: units of text (input & output)
- Claude breaks text into ‘tokens’
- Roughly speaking, ¾ of a word

Window Context
- Claude short-term memory (session memory)
- The context has a token limit

Token Limit
- Depending on model, 200K – 1M
- At >50% usage, degradation may occur

Commands
- Built-in or user defined actions
- /init, /compact, /cost, /stats, /simplify

Skills
- Advanced instructions for tasks & workflows
- Optionally Specify Tools, References, Assets

MCP
- Open standard for access to tools, data, & files
- Database, AWS, JIRA, Confluence, etc.

Hooks: custom rules to trigger actions on events
Loops: agent loops for iterative development
Sub-agents: AI assistants for executing subtasks

Designing Prompts

- Style 1: Tell Claude what to do
  - Give it specific pre-determined tasks
    - Add unit tests for a feature
    - Given OpenAPI spec, create Python client to fetch data hourly
  - Use formal SWE language as needed
- Style 2: Describe problem; ask Claude for possible solutions
  - Use language appropriate for the problem: formal SWE language not needed
    - Users need to find which routes are viable given weather and airspace constraints
      - You can find weather data @URL
      - You can get routes from xyz service
    - Users need to summarize the day's performance post-ops
      - Propose metrics, programming languages, and architecture
      - Evaluate alternative designs and programming languages and critique
      - Find additional information @abc
- Use natural language if unambiguous

See also: Claude Code–Prompt Engineering

Demo

Claude Sessions

- `/init`: Claude scans your workspace → creates `CLAUDE.md` file
- Set and review permissions
- Set context and define skills → modify `CLAUDE.md` file
- Prompt Claude
- Review and iterate

Demo 1: JSON data file → REST Server

NAMS-2
NASA Academic Mission Services Contract

`/init`

Permissions Prompt	Generate permissions and save to the project local settings file. Allow running python commands and bash commands to start applications, create, read, and write files. Deny access to environment files and other sensitive files. Ask me for installations, web access, and other major decisions.
Task Prompt	Use @fuser.json as mock data to generate a Fast API server that exposes APIs, enables users to query on fields like airports, icao ids, etc.. Provide instructions to run the server. Provide samples with HTTP GET/POST methods to demonstrate API response from mockup data.

CROWN
AIR MOBILITY. ADVANCED.

Demo 2: Reverse Engineer REST API

Start Fresh	`/clear`
	`! rm -rf .claude`
	`! rm claud.md`
	`! rm *.yaml`
	`/init`

Permissions Prompt	Generate permissions and save to the project local settings file. Allow running python commands and bash commands to start applications, create, read, and write files. Deny access to environment files and other sensitive files. Ask me for installations, web access, and other major decisions.
Task Prompt	Generate a REST API spec in YAML.

Demo 3: Documentation

Generate a document describing the service design. Generate data flow diagrams and list dependencies and services/APIs used by the service. Identify configuration parameters and potential configuration parameters.

Demo 4: Requirements

Generate a set of requirements for a service that implements the REST API subject to the following. Assume the service will be made available nationwide and will field many concurrent requests. Include devops requirements. Further assume that the flight data will be acquired from a high-performance SQL database (e.g., postgresql). Extend the API to allow the caller to specify flights to include or exclude in the response based on flight data fields such as origin and destination. Also allow for inclusion or exclusion of flight data fields in the response.

Demo 5: Design

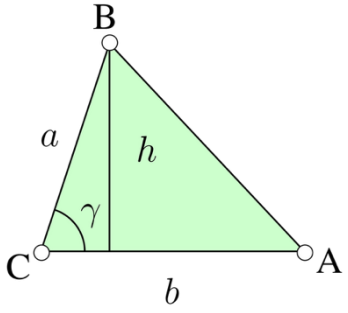
Given the generated requirements, propose an architecture and design for the service and provide one or more alternative designs and programming languages. Critique the alternatives.

Demo 6: Plan

```
/plan
```

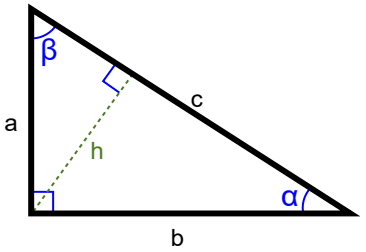
Given the generated requirements and assuming Python3 and a team of two advanced Python developers, create a project plan with stories and tasks for biweekly agile sprints. Assume you will generate the code but that the developers will review, test, and make minor adjustments to the project requirements and constraints. Assume one or two iterations per task. Provide task estimates and total project duration along with estimates of your costs. Suggest retrospective topics and frequency.

Demo 7: Reasoning – Trick Question



`/clear`

Given the right-angle triangle in @../right-triangle.svg with base c = 9 and height h = 7. Compute the area of the triangle.



Given the triangle in @../triangle.png, with base b = 9, compute the maximum height such that there exists a position for vertex B such that the angle BA to BC is a right-angle.

`/clear`

Given the right-angle triangle in @right-triangle.svg with base c = 9 and height h = 7. Check assumptions and compute the area of the triangle.

Given the right-angle triangle in @right-triangle.svg with base c = 9 and height h = 7. Check for and report errors in problem statement then compute the area of the triangle.

A better Prompt

Concluding Remarks

Working with Claude Code Effectively

- Provide concise context for the work
  - Point to documents, codebase, and assets ***essential*** for the project
  - Point to coding standards
- Scope tasks: smaller well-defined tasks
- Define essential skills to facilitate collaboration with Claude
  - Interview, executive summary, development plan, implementation, exploration/simplification
  - Require Claude to report its work prior to implementation
- Verification
  - Require proof via logs, results, tests etc.
  - Specify tests to write and run
- Compact the conversation (/compact or /clear)

In Conclusion

- Claude Code is a capable co-developer
  - As a low-level developer
    - User performs design and task breakdown
    - User assigns coding tasks to Claude
  - As a mid-level developer
    - User collaborates with Claude to create executive summaries and high-level designs
    - User collaborates with Claude to generate requirements and create task plans
    - User assigns coding tasks to Claude
- User must review Claude work
- User will iterate with Claude

Reference

Installation & Setup

- Submit the NAMS request 'NMC AI Hub’
  - Select 'User' Role
  - Optionally select 'Use with Code Assistant'
  - Paid Usage Options
    - First, select 'Project Funding Pool' and then select 'Adding yourself to an existing project pool'
    - **Funding Project Name**: ATMS
    - **MCP Quote Number**: Q - 0488
    - **Funding Project RA POC (Email)**: jane.marin@nasa.gov
  - Add a description about how you will use the LLM API Portal

Installation & Setup

- Once approved (you receive an invite to create an account with 'LiteLLM')
  - Navigate to: https://proxy.fast.luna.nasa.gov/ui/
  - Create your account with username and password, and login
- Create an API Key
  - Within the LiteLLM Interface, navigate to 'Virtual Keys'
  - Click '+Create New Key'
  - Select the ATMS team
  - Select 'All Team Models'
  - Add optional settings for your use case if any
  - Select 'Create Key'
  - COPY & SAVE your key: you will need it later

Installation & Setup

- Install Claude Code
  - See Claude Code Docs: Getting Started
- Use your key!
  - Setup environment variables
    - `export ANTHROPIC_BASE_URL=https://proxy.fast.luna.nasa.gov`
    - `export ANTHROPIC_AUTH_TOKEN="your-team-key”`
    - `export CLAUDE_CODE_DISABLE_EXPERIMENTAL_BETAS=1`
  - Start Claude in a terminal window in a trusted directory
    - `claude --model claude-sonnet-4.5`
- IDE integration supported (see GSFC-AI on Sharepoint)
  - VSCode, JetBrains, Eclipse

Models

- Commercial Claude models (non-ITAR)
  - Mythos 5: high-risk domain capability, not available publicly
  - Fable 5: suspended access by US government directive
    - Capabilities: best overall for coding, reasoning, agentic tasks, highest cost
    - Domains: coding, molecular biology, finance, cybersecurity, advanced vision, deep knowledge, multimodal, spatial reasoning
    - Restrictions: enforced limits on high-risk domains like biosecurity, cybersecurity, etc.
  - Opus 4.8
    - Capabilities: coding, complex problem-solving (reasoning), agentic tasks, higher cost
    - Domains: coding, life sciences, finance, cybersecurity, vision
  - Sonnet 4.6
    - Capabilities: coding, agentic, good performance vs. cost
    - Domains: coding, finance, and cybersecurity
  - Haiku 4.5
    - Capabilities: coding, speed, cost-efficient
    - Domains: general knowledge, document processing

Tokens and Token Limits

- Tokens: used to process text (read, generate, etc.)
  - Cumulative: entire conversation history + current prompt
  - Input and output tokens have different cost
- Context window
  - Token limit
    - Opus, Sonnet: 500K tokens (~1300 pages)
    - Haiku: 200K tokens (~500 pages)
  - Overuse leads to degraded performance and eventually hard stop
    - At ~50% limit, models start to lose accuracy or get confused (context rot)
    - More information is not necessarily better

CLAUDE.md

- Created by Claude, modified by user
  - Claude summarizes the current conversation in the file
  - Loaded on startup (a briefing of sorts)
  - Provides context for new sessions

Working with Claude

- Don’t [have to] write code or decompose tasks or generate requirements or plan agile sprints or create documents
  - Delegate these tasks to Claude
    - Claude can work with collaboration tools, version control, etc. (see MCP slide later)
  - Review results, check assumptions, rerun tests, etc.
  - Iterate with Claude
- Adjust your task estimates accordingly
  - Claude can help with that!
- To improve behavior (determinism)
  - Provide constraints
  - Create spec
  - Focus on small tasks

NOTE: the models are non-deterministic; same prompt does not mean same result

Claude Commands – Built-in

- **/init**: scans your directory to generate a CLAUDE.md file
- **/plan**: activates plan mode; Claude will submit proposal/changes and ask if it should proceed with implementation
- **/cost**: dollars spent, tokens used
- **/stats**: usage statistics last week, month, or all time
- **/compact**: reduce token usage
- **/model**: select model
- **/tasks:** check background tasks
- **/code-review ultra, /review [PR]**
- **/simplify:** review code and cleanup (w/o bug finding) (same as **/code-review --fix**)
- **@** reference a file or directory
- **!** run a terminal command
- **&** run a task in the background

Claude Basics

- /permissions: set permissions manually
- CLI Flags
  - Resume last conversation (--resume or /resume within session)
  - Use model (--model or /model within session)
  - Enable verbose mode (--verbose) * increased token usage
- Hooks: custom rules to trigger actions on events
- Commands: repetitive custom tasks
- Skills: text-based packages that teach how to handle tasks, workflows, etc.
- MCP: provide access to data and tools

Claude Built-in Tools

- Read
- Write
- Bash
- Cron
- Web tools
- Adobe
- Images
- To add a custom tool, connect an MCP server (see MCP slide)

Skills

- Used to extend Claude capability
- Define workflows, examples, patterns, things to look out for, etc.
- Used to accomplish a common task
- Example: `interview` skill
  - Interview me about all aspects till we converge on a common understanding
  - Consider all design steps and resolve dependencies
  - Where possible, explore the codebase to find answers
- Front matter: e.g., invoke automatically

Skills

- Directory Structure
  - `~/.claude/skills/` or `.claude/skills/`
    - `skill-name/`
      - `SKILL.md` (required)
      - `Scripts/` (Python, bash, etc: optional)
      - `example.md, examples/` (usage: optional)
      - `references.md, references/` (docs: optional)
      - `assets/` (data: optional)
- Invocation: `/skill-name`

Skills – Example

```
---
name: deploy-app
description: Deploy the application to production
disable-model-invocation: true
---

Deploy $ARGUMENTS to production:

1. Run the test suite
2. Build the application
3. Push to the deployment target
4. Verify the deployment succeeded
```

Source: Claude Skills Documentation

Model Context Protocol (MCP)

- Claude has a wide knowledge base, but it may also need additional information, tools, and services to accomplish its tasks
- MCP is an open-source standard designed to address this need
- MCP connects Claude to external tools, databases, and files
  - SQL, NoSQL databases
  - Atlassian Jira, Confluence
  - Git, Github
  - AWS
  - Postman
  - SMTP

MCP

- Postgres example (bash)

```
claude mcp add postgres -- npx -y @modelcontextprotocol/server-
postgres "postgresql://USER:PASSWORD@HOST:PORT/DBNAME"
```

- Restart
- Test; e.g.,
  - List database tables
  - Write query for flights between 8AM and 12PM zulu

Prompt Engineering

- See the Claude Code–Prompt Engineering page:
  - Claude responds well to clear, explicit instructions. Being specific about your desired output can help enhance results. If you want "above and beyond" behavior, explicitly request it rather than relying on the model to infer this from vague prompts.
  - Think of Claude as a brilliant but new employee who lacks context on your norms and workflows. The more precisely you explain what you want, the better the result.
  - Golden rule: Show your prompt to a colleague with minimal context on the task and ask them to follow it. If they'd be confused, Claude will be too.
    - Be specific about the desired output format and constraints.
    - Provide instructions as sequential steps using numbered lists or bullet points when the order or completeness of steps matters.