Complete GitHub Repository Structure

File Mapping from Downloaded Artifacts

Here's how to rename and organize your downloaded files:

Downloaded Files → Repository Files

```
metagenome_pipeline.groovy        → main.nf
nextflow_config.groovy            → nextflow.config
base_config.groovy                → conf/base.config
slurm_config.groovy               → conf/slurm.config
aws_config.groovy                 → conf/awsbatch.config
docker_singularity_config.groovy  → conf/docker.config
kneaddata_module.groovy           → modules/preprocessing/kneaddata.nf
metaphlan_module.groovy           → modules/taxonomy/metaphlan.nf
humann_module.groovy              → modules/functional/humann.nf
megahit_module.groovy             → modules/assembly/megahit.nf
binning_modules.groovy            → modules/binning/metabat2.nf
annotation_modules.groovy         → modules/annotation/prodigal.nf
mapping_modules.groovy            → modules/mapping/bwa.nf
growth_cdhit_modules.groovy       → modules/growth/demic.nf
readme_file.txt                   → README.md
setup_guide.txt                   → SETUP.md
project_structure.txt             → PROJECT_STRUCTURE.md
usage_examples.txt                → EXAMPLES.md
getting_started.txt               → GETTING_STARTED.md
example_samplesheet.csv           → samplesheet.csv
quick_start_script.sh             → scripts/run_pipeline.sh
```

Complete Repository Structure

```
metagenomics-nf-pipeline/
│
├── main.nf                          # Main workflow file
├── nextflow.config                  # Main configuration file
├── README.md                        # Main documentation
├── SETUP.md                         # Installation guide
├── GETTING_STARTED.md               # Quick start guide
├── EXAMPLES.md                      # Usage examples
│
├── PROJECT_STRUCTURE.md             # This structure document
```

```
├── PROJECT_STRUCTURE.md          # This structure document
├── CHANGELOG.md                  # Version history
├── LICENSE                       # MIT License
├── .gitignore                    # Git ignore file
├── .github/                      # GitHub specific files
│   ├── workflows/
│   │   └── ci.yml                # CI/CD workflow
│   ├── ISSUE_TEMPLATE/
│   │   ├── bug_report.md
│   │   └── feature_request.md
│   └── pull_request_template.md
│
├── conf/                         # Configuration files
│   ├── base.config               # Base process configuration
│   ├── docker.config             # Docker configuration
│   ├── singularity.config        # Singularity configuration
│   ├── conda.config              # Conda configuration
│   ├── slurm.config              # SLURM HPC configuration
│   ├── awsbatch.config           # AWS Batch configuration
│   ├── google.config             # Google Cloud configuration
│   └── test.config               # Test dataset configuration
│
├── modules/                      # Nextflow process modules
│   ├── qc/
│   │   ├── fastqc.nf             # FastQC quality control
│   │   └── multiqc.nf            # MultiQC aggregation
│   │
│   ├── preprocessing/
│   │   └── kneaddata.nf          # KneadData preprocessing
│   │
│   ├── taxonomy/
│   │   └── metaphlan.nf          # MetaPhlAn taxonomic profiling
│   │
│   ├── functional/
│   │   └── humann.nf             # HUMAnN functional profiling
│   │
│   ├── assembly/
│   │   ├── megahit.nf            # MEGAHIT assembler
│   │   ├── spades.nf             # SPAdes assembler
│   │   └── filter_contigs.nf     # Contig filtering
│   │
│   ├── annotation/
│   │   ├── prodigal.nf           # Prodigal gene prediction
│   │   ├── kegg.nf               # KEGG annotation
│   │   ├── cazy.nf               # CAZy annotation
│   │   └── ardb.nf               # ARDB antibiotic resistance
```

```
│   │   ├── ardb.nf              # ARDB antibiotic resistance
│   │   │
│   │   ├── mapping/
│   │   │   ├── bwa.nf           # BWA alignment
│   │   │   ├── bowtie2.nf       # Bowtie2 alignment
│   │   │   └── samtools.nf      # SAMtools operations
│   │   │
│   │   ├── binning/
│   │   │   ├── metabat2.nf      # MetaBAT2 binning
│   │   │   ├── maxbin2.nf       # MaxBin2 binning
│   │   │   ├── concoct.nf       # CONCOCT binning
│   │   │   ├── dastool.nf       # DAS_Tool integration
│   │   │   └── checkm.nf        # CheckM quality assessment
│   │   │
│   │   ├── clustering/
│   │   │   └── cdhit.nf         # CD-HIT clustering
│   │   │
│   │   └── growth/
│   │       └── demic.nf         # DEMIC growth rates
│   │
│   ├── bin/                     # Executable scripts
│   │   ├── filter_bins_quality.py    # Filter bins by quality
│   │   ├── parse_kegg_results.py     # Parse KEGG annotations
│   │   ├── parse_cazy_results.py     # Parse CAZy annotations
│   │   ├── summarize_taxonomy.R      # Summarize taxonomy
│   │   ├── summarize_functions.R     # Summarize functions
│   │   ├── plot_growth_rates.R       # Plot growth rates
│   │   └── merge_metaphlan_tables.py # Merge MetaPhlAn outputs
│   │
│   ├── scripts/                 # Helper scripts
│   │   ├── run_pipeline.sh           # Interactive pipeline launcher
│   │   ├── setup_databases.sh        # Database download script
│   │   ├── validate_installation.sh  # Check dependencies
│   │   ├── generate_samplesheet.py   # Generate samplesheet
│   │   ├── clean_workdir.sh          # Clean work directory
│   │   └── download_test_data.sh     # Download test dataset
│   │
│   ├── assets/                  # Static assets
│   │   ├── samplesheet_schema.json   # Input validation schema
│   │   ├── multiqc_config.yaml       # MultiQC configuration
│   │   └── logo.png                  # Pipeline logo
│   │
│   ├── docs/                    # Additional documentation
│   │   ├── usage.md                  # Detailed usage guide
│   │   ├── output.md                 # Output description
│   │   ├── parameters.md             # Parameter reference
```

```
│   ├── troubleshooting.md          # Troubleshooting guide
│   ├── databases.md                # Database setup details
│   ├── containers.md               # Container information
│   └── citation.md                 # How to cite
│
├── test/                           # Test data and configuration
│   ├── test_samplesheet.csv        # Test sample sheet
│   ├── test.config                 # Test configuration
│   ├── README.md                   # Test documentation
│   └── data/                       # Small test datasets
│       └── .gitkeep                # Keep empty directory
│
├── workflows/                      # Sub-workflows (optional)
│   ├── taxonomy_profiling.nf       # Taxonomy sub-workflow
│   ├── functional_profiling.nf     # Function sub-workflow
│   ├── assembly_binning.nf         # Assembly sub-workflow
│   └── README.md                   # Workflow documentation
│
└── examples/                       # Example files
    ├── samplesheet.csv             # Example samplesheet
    ├── samplesheet_coassembly.csv  # Co-assembly example
    ├── run_local.sh                # Local run example
    ├── run_hpc.sh                  # HPC run example
    └── run_aws.sh                  # AWS run example
```

Step-by-Step Repository Setup

Step 1: Create Repository Structure

bash

```bash
# Create main directory
mkdir metagenomics-nf-pipeline
cd metagenomics-nf-pipeline

# Create all subdirectories
mkdir -p conf
mkdir -p modules/{qc,preprocessing,taxonomy,functional,assembly,annotation,mapping,binning,clustering,growth}
mkdir -p bin
mkdir -p scripts
mkdir -p assets
mkdir -p docs
mkdir -p test/data
mkdir -p workflows
mkdir -p examples
mkdir -p .github/{workflows,ISSUE_TEMPLATE}

# Create placeholder files
touch .gitkeep test/data/.gitkeep
```

Step 2: Move Downloaded Files

bash

```
# Assuming downloaded files are in ~/Downloads/

# Main files (rename .groovy to .nf or appropriate extension)
mv ~/Downloads/metagenome_pipeline.groovy main.nf
mv ~/Downloads/nextflow_config.groovy nextflow.config

# Configuration files
mv ~/Downloads/base_config.groovy conf/base.config
mv ~/Downloads/slurm_config.groovy conf/slurm.config
mv ~/Downloads/aws_config.groovy conf/awsbatch.config
mv ~/Downloads/docker_singularity_config.groovy conf/docker.config

# Module files
mv ~/Downloads/kneaddata_module.groovy modules/preprocessing/kneaddata.nf
mv ~/Downloads/metaphlan_module.groovy modules/taxonomy/metaphlan.nf
mv ~/Downloads/humann_module.groovy modules/functional/humann.nf
mv ~/Downloads/megahit_module.groovy modules/assembly/megahit.nf

# Split combined module files
# binning_modules.groovy contains multiple processes
# You'll need to split this into separate files:
# - metabat2.nf, maxbin2.nf, concoct.nf, dastool.nf, checkm.nf

# Similarly for other combined modules
mv ~/Downloads/annotation_modules.groovy modules/annotation/prodigal.nf
mv ~/Downloads/mapping_modules.groovy modules/mapping/bwa.nf
mv ~/Downloads/growth_cdhit_modules.groovy modules/growth/demic.nf

# Documentation files
mv ~/Downloads/readme_file.txt README.md
mv ~/Downloads/setup_guide.txt SETUP.md
mv ~/Downloads/getting_started.txt GETTING_STARTED.md
mv ~/Downloads/usage_examples.txt EXAMPLES.md
```

```
mv ~/Downloads/project_structure.txt PROJECT_STRUCTURE.md

# Scripts
mv ~/Downloads/quick_start_script.sh scripts/run_pipeline.sh
chmod +x scripts/run_pipeline.sh

# Example files
mv ~/Downloads/example_samplesheet.csv examples/samplesheet.csv
```

Step 3: Split Combined Module Files

Some downloaded files contain multiple processes. Split them:

For binning_modules.groovy:

```
bash

# Create individual files for each binning tool
# Extract MetaBAT2 process → modules/binning/metabat2.nf
# Extract MaxBin2 process → modules/binning/maxbin2.nf
# Extract CONCOCT process → modules/binning/concoct.nf
# Extract DAS_Tool process → modules/binning/dastool.nf
# Extract CheckM process → modules/binning/checkm.nf
```

For annotation_modules.groovy:

```
bash

# Extract Prodigal process → modules/annotation/prodigal.nf
# Extract KEGG process → modules/annotation/kegg.nf
# Extract CAZy process → modules/annotation/cazy.nf
# Extract filter_contigs process → modules/assembly/filter_contigs.nf
```

For mapping_modules.groovy:

```
bash

# Extract BWA processes → modules/mapping/bwa.nf
# Extract Bowtie2 processes → modules/mapping/bowtie2.nf
# Extract SAMtools processes → modules/mapping/samtools.nf
```

For growth_cdhit_modules.groovy:

```
bash
```

```
# Extract DEMIC process → modules/growth/demic.nf
# Extract CD-HIT process → modules/clustering/cdhit.nf
# Extract FastQC process → modules/qc/fastqc.nf
# Extract MultiQC process → modules/qc/multiqc.nf
# Extract SPAdes process → modules/assembly/spades.nf
```

Step 4: Create Missing Module Files

Create stub files for modules that need to be extracted:

bash

```
# QC modules
touch modules/qc/fastqc.nf
touch modules/qc/multiqc.nf

# Additional assembly
touch modules/assembly/spades.nf

# Additional annotation
touch modules/annotation/kegg.nf
touch modules/annotation/cazy.nf
touch modules/annotation/ardb.nf

# Mapping
touch modules/mapping/bowtie2.nf
touch modules/mapping/samtools.nf

# Binning
touch modules/binning/maxbin2.nf
touch modules/binning/concoct.nf
touch modules/binning/dastool.nf
touch modules/binning/checkm.nf

# Clustering
touch modules/clustering/cdhit.nf
```

Step 5: Create Essential Configuration Files

Create conf/singularity.config:

bash

```
cat > conf/singularity.config << 'EOF'
/*
```

```
================================================================================
  Singularity Configuration
================================================================================
*/

singularity {
    enabled = true
    autoMounts = true
    cacheDir = "$HOME/.singularity_cache"
}

process {
    // Container settings for each process are defined in the module files
}
EOF
```

Create conf/conda.config:

```
bash




cat > conf/conda.config << 'EOF'
/*
================================================================================
  Conda Configuration
================================================================================
*/

conda {
    enabled = true
    cacheDir = "$HOME/.conda_cache"
}

process {
    // Conda environment settings defined in module files
}
EOF
```

Create conf/google.config:

bash

```
cat > conf/google.config << 'EOF'
/*
========================================================================================
  Google Cloud Configuration
========================================================================================
*/

process {
    executor = 'google-lifesciences'
}

google {
    region = 'us-central1'
    project = 'your-project-id'
}
EOF
```

Create conf/test.config:

bash

```
cat > conf/test.config << 'EOF'
/*
========================================================================================
  Test Configuration
========================================================================================
*/

params {
    // Test dataset parameters
    input = 'test/test_samplesheet.csv'
    outdir = 'test_results'
    max_cpus = 2
```

```
    max_memory = 6.GB
    max_time = 2.h
}
EOF
```

Step 6: Create Additional Essential Files

Create .gitignore:

bash

```
cat > .gitignore << 'EOF'
# Nextflow
work/
.nextflow/
.nextflow.log*
*.html
*.pdf
trace.txt
timeline.html
report.html
dag.dot

# Results
results/
test_results/

# Databases (too large for git)
databases/
*.dmnd
*.bt2
*.bt2l
*.bam
*.sam
*.fasta.gz
*.fastq.gz

# Python
__pycache__/
*.py[cod]
*$py.class

.Python
*.so
.pytest_cache/
```

```
# R
.Rhistory
.RData
.Rproj.user

# IDEs
.vscode/
.idea/
*.swp
*.swo
*~

# System
.DS_Store
Thumbs.db

# Singularity
*.sif

# Conda
.conda_cache/

# Temporary files
*.tmp
*.temp
EOF
```

Create LICENSE:

```
bash









cat > LICENSE << 'EOF'
MIT License
```

```
Copyright (c) 2024 Ashok K. Sharma

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
EOF
```

Create CHANGELOG.md:

```
bash
```

```
cat > CHANGELOG.md << 'EOF'
# Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.0.0/),
and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.html).

## [1.0.0] - 2024-10-06

### Added
- Initial release of comprehensive metagenomic analysis pipeline
- Quality control with FastQC and MultiQC
- Read preprocessing with KneadData
- Taxonomic profiling with MetaPhlAn4
- Functional profiling with HUMAnN3
- Metagenomic assembly with MEGAHIT and SPAdes
- Gene prediction with Prodigal
- Genome binning with MetaBAT2, MaxBin2, and CONCOCT
- Bin integration with DAS_Tool
- Bin quality assessment with CheckM
- Bacterial growth rate calculation with DEMIC
- Functional annotation with KEGG and CAZy
- Support for Docker, Singularity, and Conda
- Support for SLURM HPC and AWS Batch execution
- Comprehensive documentation and examples

### Features
- Modular design allowing selective step execution
- Individual and co-assembly strategies
- Multiple binning tool integration
- Automatic error handling and retry logic
- Resume capability for interrupted runs
- Resource optimization with dynamic scaling

[1.0.0]: https://github.com/yourusername/metagenomics-nf-pipeline/releases/tag/v1.0.0
EOF
```

Step 7: Create GitHub-specific Files

.github/workflows/ci.yml:

bash

```
mkdir -p .github/workflows
cat > .github/workflows/ci.yml << 'EOF'
name: CI

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main ]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Install Nextflow
        run: |
          wget -qO- https://get.nextflow.io | bash
          sudo mv nextflow /usr/local/bin/
      - name: Lint Nextflow code
        run: nextflow run main.nf --help

  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Install Nextflow
        run: |
          wget -qO- https://get.nextflow.io | bash
          sudo mv nextflow /usr/local/bin/
      - name: Run pipeline test
        run: |
          # Add test command here when test data is available
          echo "Test pipeline execution"
EOF
```

```
EOF
```

.github/ISSUE_TEMPLATE/bug_report.md:

```
bash
```

```
mkdir -p .github/ISSUE_TEMPLATE
cat > .github/ISSUE_TEMPLATE/bug_report.md << 'EOF'
---
name: Bug report
about: Create a report to help us improve
title: '[BUG] '
labels: bug
assignees: ''
---

**Describe the bug**
A clear and concise description of what the bug is.

**To Reproduce**
Steps to reproduce the behavior:
1. Command run: '...'
2. Parameters used: '...'
3. Error message: '...'

**Expected behavior**
A clear and concise description of what you expected to happen.

**Environment:**
 - OS: [e.g. Ubuntu 20.04]
 - Nextflow version: [e.g. 21.10.0]
 - Execution profile: [e.g. docker, singularity]
 - Pipeline version: [e.g. 1.0.0]

**Additional context**
Add any other context about the problem here.
EOF
```

.github/ISSUE_TEMPLATE/feature_request.md:

bash

```
cat > .github/ISSUE_TEMPLATE/feature_request.md << 'EOF'
---
name: Feature request
about: Suggest an idea for this project
title: '[FEATURE] '
labels: enhancement
assignees: ''
---

**Is your feature request related to a problem? Please describe.**
A clear and concise description of what the problem is.

**Describe the solution you'd like**
A clear and concise description of what you want to happen.

**Describe alternatives you've considered**
A clear and concise description of any alternative solutions or features you've considered.

**Additional context**
Add any other context or screenshots about the feature request here.
EOF
```

.github/pull_request_template.md:

.github/pull_request_template.md:

```bash
cat > .github/pull_request_template.md << 'EOF'
## Description
Please include a summary of the change and which issue is fixed.

Fixes # (issue)

## Type of change
- [ ] Bug fix
- [ ] New feature
- [ ] Documentation update
- [ ] Performance improvement

## Checklist:
- [ ] My code follows the style guidelines of this project
- [ ] I have performed a self-review of my own code
- [ ] I have commented my code, particularly in hard-to-understand areas
- [ ] I have made corresponding changes to the documentation
- [ ] My changes generate no new warnings
- [ ] I have added tests that prove my fix is effective or that my feature works
- [ ] New and existing unit tests pass locally with my changes
EOF
```

Step 8: Create Helper Scripts

scripts/setup_databases.sh:

```bash
cat > scripts/setup_databases.sh << 'EOF'
#!/bin/bash

# Database setup script
# This script downloads all required databases

set -e

DB_DIR="${1:-$HOME/metagenomics_databases}"
```

```
echo "Setting up databases in: $DB_DIR"
mkdir -p "$DB_DIR"
cd "$DB_DIR"

# MetaPhlAn
echo "Downloading MetaPhlAn database..."
mkdir -p metaphlan_db
metaphlan --install --bowtie2db metaphlan_db

# HUMAnN
echo "Downloading HUMAnN databases..."
mkdir -p humann_dbs
humann_databases --download chocophlan full humann_dbs
humann_databases --download uniref uniref90_diamond humann_dbs

# CheckM
echo "Downloading CheckM database..."
mkdir -p checkm_data
cd checkm_data
wget https://data.ace.uq.edu.au/public/CheckM_databases/checkm_data_2015_01_16.tar.gz
tar -xzf checkm_data_2015_01_16.tar.gz
checkm data setRoot $(pwd)
cd ..

echo "Database setup complete!"
echo "Database location: $DB_DIR"
EOF

chmod +x scripts/setup_databases.sh
```

scripts/validate_installation.sh:

```
bash
```

```
cat > scripts/validate_installation.sh << 'EOF'
#!/bin/bash

# Validate installation script

echo "Checking installation..."

# Check Nextflow
if command -v nextflow &> /dev/null; then
    echo "✓ Nextflow installed: $(nextflow -version | head -n1)"
else
    echo "✗ Nextflow not found"
fi

# Check Docker
if command -v docker &> /dev/null; then
    echo "✓ Docker installed: $(docker --version)"
else
```

```
    echo "✗ Docker not found"
fi

# Check Singularity
if command -v singularity &> /dev/null; then
    echo "✓ Singularity installed: $(singularity --version)"
else
    echo "✗ Singularity not found"
fi

# Check Conda
if command -v conda &> /dev/null; then
    echo "✓ Conda installed: $(conda --version)"
else
    echo "✗ Conda not found"
fi

echo "Validation complete!"
EOF

chmod +x scripts/validate_installation.sh
```

scripts/generate_samplesheet.py:

```
bash

cat > scripts/generate_samplesheet.py << 'EOF'
#!/usr/bin/env python3
"""
Generate samplesheet from directory of FASTQ files
"""

import os
import sys
import argparse
import re
from pathlib import Path

def find_fastq_pairs(directory, pattern):
    """Find paired FASTQ files in directory"""
    fastq_files = {}

    for file in Path(directory).glob(pattern):
        filename = file.name
```

```
        # Extract sample name (everything before _R1 or _R2)
        match = re.search(r'(.+?)_R([12])', filename)
        if match:
            sample = match.group(1)
            read = match.group(2)

            if sample not in fastq_files:
                fastq_files[sample] = {}
            fastq_files[sample][f'R{read}'] = str(file.absolute())

    return fastq_files

def write_samplesheet(fastq_files, output):
    """Write samplesheet CSV"""
    with open(output, 'w') as f:
        f.write('sample,fastq_1,fastq_2\n')
        for sample, reads in sorted(fastq_files.items()):
            if 'R1' in reads and 'R2' in reads:
                f.write(f'{sample},{reads["R1"]},{reads["R2"]}\n')
            else:
                print(f"Warning: Incomplete pair for {sample}", file=sys.stderr)

def main():
    parser = argparse.ArgumentParser(description='Generate samplesheet from FASTQ directory')
    parser.add_argument('--directory', required=True, help='Directory containing FASTQ files')
    parser.add_argument('--output', required=True, help='Output samplesheet CSV')
    parser.add_argument('--pattern', default='*_R*.fastq.gz', help='FASTQ file pattern')

    args = parser.parse_args()

    fastq_files = find_fastq_pairs(args.directory, args.pattern)
    write_samplesheet(fastq_files, args.output)

    print(f"Generated samplesheet with {len(fastq_files)} samples: {args.output}")

if __name__ == '__main__':
    main()
EOF

chmod +x scripts/generate_samplesheet.py
```

scripts/clean_workdir.sh:

```
bash
```

```
cat > scripts/clean_workdir.sh << 'EOF'
#!/bin/bash

# Clean work directory script

WORK_DIR="${1:-work}"

echo "Cleaning work directory: $WORK_DIR"

if [ -d "$WORK_DIR" ]; then
    du -sh "$WORK_DIR"
    read -p "Are you sure you want to delete? (yes/no): " confirm
    if [ "$confirm" = "yes" ]; then
        rm -rf "$WORK_DIR"
        echo "Work directory cleaned"
    else
        echo "Cancelled"
    fi
else
    echo "Work directory not found: $WORK_DIR"
fi
EOF

chmod +x scripts/clean_workdir.sh
```

Step 9: Create Example Files

examples/run_local.sh:

```
bash
```

```
cat > examples/run_local.sh << 'EOF'
#!/bin/bash

# Example: Run locally with Docker

nextflow run ../main.nf \
    --input samplesheet.csv \
    --outdir results \
    --host_genome ~/databases/human_genome/human_GRCh38 \
    --metaphlan_db ~/databases/metaphlan_db \
    --humann_nucleotide_db ~/databases/humann_dbs/chocophlan \
    --humann_protein_db ~/databases/humann_dbs/uniref \
    -profile docker \
    -resume
EOF

chmod +x examples/run_local.sh
```

examples/run_hpc.sh:

bash

```
cat > examples/run_hpc.sh << 'EOF'
#!/bin/bash
#SBATCH --job-name=metagenomics
#SBATCH --time=72:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G
#SBATCH --output=pipeline_%j.log

module load singularity nextflow

nextflow run /path/to/metagenomics-nf-pipeline/main.nf \
    --input samplesheet.csv \
    --outdir results \
    --host_genome /data/databases/human_genome/human_GRCh38 \
    --metaphlan_db /data/databases/metaphlan_db \
    --humann_nucleotide_db /data/databases/humann_dbs/chocophlan \
    --humann_protein_db /data/databases/humann_dbs/uniref \
    -profile slurm \
    -resume
EOF
```