Tutorial 2: Basic STUNIR Workflow

Duration: 15 minutes
Level: Beginner
Prerequisites: Completed Tutorial 1

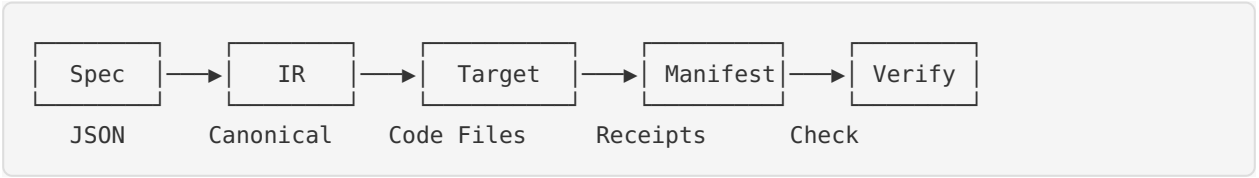
Video Script

Introduction (0:00 - 0:45)

> "Welcome back! In this tutorial, we'll walk through the complete STUNIR workflow - from spec to verified code. You'll learn about each stage of the pipeline and how they work together."

The STUNIR Pipeline (0:45 - 2:00)

Show diagram:



> "The pipeline has five stages: Spec input, IR generation, target code emission, manifest creation, and verification. Each stage is deterministic."

Stage 1: Creating a Spec (2:00 - 4:00)

Create `calculator.spec.json`:

```
{
  "name": "calculator",
  "version": "1.0.0",
  "description": "A simple calculator module",
  "functions": [
    {
      "name": "add",
      "params": [
        {"name": "a", "type": "i32"},
        {"name": "b", "type": "i32"}
      ],
      "returns": "i32",
      "description": "Add two integers"
    },
    {
      "name": "divide",
      "params": [
        {"name": "x", "type": "f64"},
        {"name": "y", "type": "f64"}
      ],
      "returns": "f64",
      "description": "Divide two floats"
    }
  ],
  "exports": ["add", "divide"]
}
```

Best practices:

> "Always include version numbers, descriptions, and explicit exports. This makes your spec self-documenting."

Stage 2: IR Generation (4:00 - 6:00)

Terminal Demo:

```
# Create output directory
mkdir -p output/ir

# Generate IR
python3 tools/ir_emitter/emit_ir.py calculator.spec.json output/ir/calculator.ir.json

# Examine the IR
python3 -m json.tool output/ir/calculator.ir.json
```

Explain IR structure:

- `ir_version` : IR format version
- `ir_epoch` : Generation timestamp
- `ir_spec_hash` : Hash of input spec
- `module` : Module metadata
- `functions` : Normalized function definitions

Stage 3: Code Generation (6:00 - 9:00)

Generate Python code:

```
# Using the emitter
python3 tools/emitters/emit_code.py \
  --target python \
  --ir output/ir/calculator.ir.json \
  --output output/python/
```

Show output:

```
# output/python/calculator.py
"""Generated Python module: calculator
Version: 1.0.0
Generated by STUNIR
"""

def add(a: int, b: int) -> int:
    """Add two integers"""
    pass  # Implementation placeholder

def divide(x: float, y: float) -> float:
    """Divide two floats"""
    pass  # Implementation placeholder
```

Generate Rust code:

```
python3 tools/emitters/emit_code.py \
  --target rust \
  --ir output/ir/calculator.ir.json \
  --output output/rust/
```

Stage 4: Manifest Generation (9:00 - 11:00)

Terminal Demo:

```
# Generate IR manifest
python3 manifests/ir/gen_ir_manifest.py \
  --ir-dir output/ir/ \
  --output receipts/ir_manifest.json

# Generate targets manifest
python3 manifests/targets/gen_targets_manifest.py \
  --targets-dir output/ \
  --output receipts/targets_manifest.json

# View manifests
ls -la receipts/
```

Explain manifest structure:
- Schema identifier
- Generation epoch
- Entry list with hashes
- Manifest hash (integrity)

Stage 5: Verification (11:00 - 13:00)

Terminal Demo:

```
# Verify IR manifest
python3 manifests/ir/verify_ir_manifest.py receipts/ir_manifest.json

# Verify targets manifest
python3 manifests/targets/verify_targets_manifest.py receipts/targets_manifest.json

# Strict verification
./scripts/verify_strict.sh --strict
```

Expected output:

```
✅ Manifest verified: 2 entries, 0 errors
```

Complete Workflow Script (13:00 - 14:00)

Create `build.sh`:

```
#!/bin/bash
set -e

SPEC="$1"
OUTPUT="output"

echo "=== STUNIR Build ==="

# Stage 1-2: Generate IR
mkdir -p $OUTPUT/ir
python3 tools/ir_emitter/emit_ir.py "$SPEC" $OUTPUT/ir/module.ir.json
echo "✅ IR generated"

# Stage 3: Generate targets
for target in python rust; do
  mkdir -p $OUTPUT/$target
  python3 tools/emitters/emit_code.py --target $target \
    --ir $OUTPUT/ir/module.ir.json --output $OUTPUT/$target/
done
echo "✅ Targets generated"

# Stage 4: Generate manifests
mkdir -p receipts
python3 manifests/ir/gen_ir_manifest.py --ir-dir $OUTPUT/ir/
echo "✅ Manifests generated"

# Stage 5: Verify
python3 manifests/ir/verify_ir_manifest.py receipts/ir_manifest.json
echo "✅ Verification complete"
```

Wrap Up (14:00 - 15:00)

Summary:

- ✅ Created a comprehensive spec
- ✅ Generated canonical IR
- ✅ Emitted code for multiple targets
- ✅ Created manifests
- ✅ Verified all artifacts

Next steps:

> “In the next tutorial, we’ll explore advanced features like custom emitters and batch processing.”

Commands Reference

```
# Full workflow
./scripts/build.sh spec.json

# Individual stages
python3 tools/ir_emitter/emit_ir.py <spec> <output>
python3 tools/emitters/emit_code.py --target <lang> --ir <ir> --output <dir>
python3 manifests/ir/gen_ir_manifest.py --ir-dir <dir>
python3 manifests/ir/verify_ir_manifest.py <manifest>
```