> Create deploy scripts for all remaining AXIOM contracts to match my existing Hardhat pattern. The scripts must:

Write to config/addresses.json using the same saveAddress() helper format I already use.

Include constructor parameters, role grants, and console logs for each deployment.

Use the same naming format (05_deploy_*.js onward).

Import helpers with const { saveAddress } = require("./helpers/saveAddress") and const { getAddr } = require("./helpers/getAddr") when necessary.

Follow best practices for async/await, error handling, and role assignments.

Match my existing folder structure with /scripts/helpers/getAddr.js for reading from addresses.json.

Include deploy scripts for:

1. LiquidityVault

2. AXMStakingPool

3. GovernanceCouncil

4. AssetTokenizationRegistry

5. TreasuryManager

6. OracleRelay

7. NodeLicenseNFT

8. AXIOMLaunchpad

9. CrossChainBridgeAdapter

10. AXIOMAnalyticsHub

11. ComplianceAdapter

Each script must automatically grant roles (e.g., MANAGER_ROLE, FUNDER_ROLE, RELAYER_ROLE, etc.) from .env variables like ADMIN, STAKING_FUNDER, BRIDGE_RELAYER, etc.

Use Hardhat’s ethers.getContractFactory(), deploy(), waitForDeployment(), and saveAddress(). End each script with main().catch((e)=>{console.error(e);process.exit(1);});

Also generate:

A getAddr.js helper that reads addresses from config/addresses.json.

Updated package.json with "scripts" entries to deploy each module individually.

Updated .env template including keys for all parameters (ADMIN, LP_TOKEN, STAKING_FUNDER, GOV_*, TREASURY_OPERATOR, ORACLE_PUBLISHER, LAUNCHPAD_*, BRIDGE_RELAYER, and SCHEMA_*).

Use consistent console output like: console.log(\ContractName deployed at ${addr}`);`

After creating all scripts, ensure:

File names increment sequentially (05_deploy_LiquidityVault.js → 15_deploy_ComplianceAdapter.js).

Scripts import Hardhat and run standalone with --network peaq.

All scripts output to addresses.json under their network key.

Example structure:

```
scripts/
├── helpers/
│   ├── saveAddress.js
│   └── getAddr.js
├── 05_deploy_LiquidityVault.js
├── 06_deploy_AXMStakingPool.js
...
└── 15_deploy_ComplianceAdapter.js
```

Confirm that each deploy script mirrors the same deployment pattern and logging style used for previously deployed contracts (like AXM, DIDRegistry, etc.).
The output should be complete JavaScript code files ready to run inside my Replit Hardhat workspace.