Phase 2A Dependency Mapping & Execution Plan

Generated: 2025-09-26 14:46:22

Ansible Hierarchy Dependencies

Based on Ansible best practices, the consolidation must follow this dependency order:

1. **Infrastructure First** (inventory, vars, configs)
2. **Roles Second** (role definitions and dependencies)
3. **Playbooks Third** (orchestration that uses roles)
4. **Security Last** (security patches and configurations)

Target Branch Consolidation Order

1. infrastructure-consolidated (Priority: HIGHEST)

Branches to merge: 1
- `env-inventories-phase2` (779 files, 31 inventory changes)

Dependencies: None - this is the foundation
Merge Strategy: Direct merge after validation
Risk Level: LOW

2. phase-2-consolidated (Priority: HIGH)

Branches to merge: 13
- `doc-refactor/feedback-consolidation` (746 files)
- `feature-consolidated-production` (0 files - empty target)
- `feature/phase-3-4-production-ops` (438 files)
- `feature/sprint2-advanced` (107 files)
- `feature/sprint3-operational-excellence` (151 files)
- `feature/sprint4-final-production` (186 files)
- `fix-ci-pipeline` (186 files)
- `infrastructure-consolidated` (0 files - empty target)
- `phase-2-ansible-standards` (776 files)
- `phase-2-consolidated` (0 files - empty target)
- `phase-3.3-backup-automation` (502 files)
- `phase2-role-standardization` (742 files)
- `phase4/quality-standards-complete` (56 files)

Dependencies: Must wait for infrastructure-consolidated
Merge Strategy: Squash merge with archive-before-merge
Risk Level: MEDIUM (large number of branches)

3. feature-consolidated-production (Priority: MEDIUM)

Branches to merge: 5
- `feat/var-templates-phase3` (765 files)

- `feature/phase2-security` (50 files)
- `feature/pin-critical-directive` (66 files)
- `feature/repo-recovery-phase1` (649 files)
- `feature/repo-recovery-phase2` (644 files)

Dependencies: Must wait for phase-2-consolidated
Merge Strategy: Squash merge with archive-before-merge
Risk Level: MEDIUM (overlapping recovery phases)

4. security-remediation-consolidated (Priority: LOWEST)

Branches to merge: 0 (no branches currently mapped to this target)

Note: Security branches are currently in KEEP category for safety

Detailed Merge Execution Plan

Phase 2B Day 1: Infrastructure Foundation

```
# 1. Merge infrastructure branch
git checkout infrastructure-consolidated
git merge --squash origin/env-inventories-phase2
git commit -m "Consolidate: env-inventories-phase2 infrastructure changes"
git push origin infrastructure-consolidated

# 2. Create PR for infrastructure-consolidated -> main
# 3. Run full test suite
# 4. Get approval and merge
```

Phase 2B Day 2: Phase/Sprint Consolidation

```
# Merge phase branches in dependency order
git checkout phase-2-consolidated

# Large branches first (most complex)
git merge --squash origin/phase-2-ansible-standards
git commit -m "Consolidate: phase-2-ansible-standards"

git merge --squash origin/phase2-role-standardization
git commit -m "Consolidate: phase2-role-standardization"

git merge --squash origin/doc-refactor/feedback-consolidation
git commit -m "Consolidate: doc-refactor/feedback-consolidation"

# Continue with remaining branches...
```

Phase 2B Day 3: Feature Consolidation

```
# Merge feature branches
git checkout feature-consolidated-production

# Recovery phases first (potential conflicts)
git merge --squash origin/feature/repo-recovery-phase1
git commit -m "Consolidate: feature/repo-recovery-phase1"

git merge --squash origin/feature/repo-recovery-phase2
git commit -m "Consolidate: feature/repo-recovery-phase2"

# Continue with remaining features...
```

Conflict Assessment Summary

- **All branches show LOW conflict level** - excellent for consolidation
- **No HIGH or MEDIUM conflicts detected** in initial analysis
- **Most conflicts expected**: Between recovery phases (repo-recovery-phase1 vs phase2)
- **Least conflicts expected**: Sprint branches (clean separation)

Risk Mitigation Strategies

1. Archive-Before-Merge Approach

- Create archive tags for all source branches before consolidation
- Format: `archive/[branch-name]-pre-consolidation-[date]`
- Enables 100% rollback capability

2. Incremental Validation

- Run ansible-lint after each merge
- Run yamllint after each merge
- Run syntax-check after each merge
- Deploy to staging environment for validation

3. Dependency Validation

- Verify inventory changes don’t break role dependencies
- Validate role changes don’t break playbook execution
- Test playbook changes in isolated environment

Success Criteria

Technical Gates

- [ ] All ansible-lint checks pass
- [ ] All yamllint checks pass
- [ ] All syntax-check validations pass
- [ ] Staging deployment successful
- [ ] No regression in existing functionality

Process Gates

- [ ] All source branches archived with tags
- [ ] All consolidation PRs reviewed and approved
- [ ] All merge commits include proper consolidation messages
- [ ] Documentation updated with consolidation history

Rollback Plan

If consolidation fails at any stage:

1. **Immediate**: Revert target branch to pre-merge state
2. **Recovery**: Restore from archive tags
3. **Validation**: Re-run full test suite
4. **Communication**: Update stakeholders on rollback status

Next Steps for Phase 2B

1. **Setup Test Gates** (ansible-lint, yamllint, syntax-check)
2. **Create GitHub Actions workflow** for automated validation
3. **Begin infrastructure consolidation** (lowest risk)
4. **Proceed with phase consolidation** (medium risk)
5. **Complete feature consolidation** (highest complexity)

Phase 2A Status: ✅ COMPLETE
Ready for Phase 2B: ✅ YES
Rollback Capability: ✅ 100% MAINTAINED