Nice — here’s a complete, meeting-ready plan plus a concrete automation approach using **Bitbucket + Jenkins** to migrate Windows file-shares to **AWS FSx for Windows File Server**. I’ll give you:

1. A short migration overview
2. A meeting checklist of requirements & questions to gather decisions and constraints
3. A pre-migration assessment checklist
4. A migration runbook (phased) with cutover/rollback steps
5. Automation design using Bitbucket (repo layout) + Jenkins (pipeline) — with example Jenkinsfile and useful CLI snippets / Terraform hints
6. Testing & verification steps and post-migration tasks

**1) Quick overview (what we’re doing)**

Migrate existing on-prem Windows SMB file shares to **AWS FSx for Windows File Server**. Data movement is performed with **AWS DataSync** (recommended) or file copy tools (Robocopy/DFS/Rsync for specific needs). Infrastructure (FSx, networking, DataSync agent, IAM, DNS) is provisioned with IaC (Terraform / CloudFormation). Bitbucket stores code; Jenkins runs the pipeline to provision infra, create DataSync tasks, run transfers, and trigger verification.

**2) Meeting checklist — questions & items to gather (ask these in the meeting)**

Group these by category — bring answers to decide approach, SLA, and scope.

**Inventory & scope**

* Exact list of servers / file shares (UNC paths), sizes (used bytes and file count), growth rate.
* Which shares are active vs archival? (hot vs cold)
* Max file size, average file size, files >100GB? lots of small files?
* Any special file types (reparse points, symlinks, hardlinks, sparse files, alternate data streams)?

**Access, permissions & integrity**

* Are NTFS permissions / ACLs required to be preserved exactly? (Yes/No)
* Are ownership and SIDs required to remain identical? (If moving to new AD domain, how to map?)
* Are file/folder timestamps required to be preserved?
* Any applications that require file locks / open file handling?

**Networking & security**

* VPC/subnet and routing: will FSx be in same VPC as apps? Need VPN/Direct Connect/Transit Gateway?
* Required throughput / latency SLA for FSx (IOPS, throughput per second).
* Security groups, NACLs, and firewall rules for SMB and DataSync agent outbound connectivity.
* DNS integration: will FSx join existing AD or use AWS Managed AD? Which domain controllers to use?

**Authentication / AD**

* Do you have Active Directory on AWS or on-prem? Will FSx join the on-prem domain or AWS Managed AD?
* Are there trusts between ADs? (affects user SIDs and ACLs)

**Cutover / downtime & business impact**

* Acceptable downtime window for each share (immediate cutover vs phased).
* How will clients be redirected (DNS CNAME swap, DFS Namespaces, client mapping change)?
* Rollback requirements and acceptance tests (what constitutes “success” for each share)?

**Tools, automation & roles**

* Are we allowed to install DataSync agent VM on premises? Are there proxy constraints?
* Who owns the Bitbucket repo & Jenkins server? Credentials / IAM role owners?
* Who will be approvers during pipeline runs (change control)?

**Backup & compliance**

* Do we need backups of source before migration? Retention policy?
* Data residency, encryption at rest and in transit requirements, auditing.

**Testing & verification**

* A list of pilot shares (small/medium/large) to try first
* Verification plan: checksums, file counts, permissions audit

**3) Pre-migration assessment checklist (technical)**

* Inventory: gather share list, sizes, file counts, top N largest files, path depth, number of small files.
* Performance profile: peak concurrent users/reads/writes, expected IOPS and throughput.
* AD & ACL mapping: map SIDs and owners if domain changes.
* Network: ensure SMB ports allowed (TCP 445) between clients and FSx or plan alternate access method; ensure DataSync agent access to AWS (agent needs outbound to AWS endpoints).
* Storage architecture: choose FSx file system type & performance mode (SSD/HDD options depending on FSx offering).
* Backup snapshots: ensure pre-migration snapshots exist (source) for rollback.
* Create pilot plan: select 1–3 shares representing different workloads.

**4) High-level migration runbook (phases)**

**Phase 0 — Prepare**

* Provision / identify project AWS account, VPC, subnets, security groups.
* Decide AD strategy: join FSx to existing AD or use AWS Managed AD + trusts.
* Create service IAM role(s) for Terraform/Jenkins and DataSync.
* Create Bitbucket repo and branch structure (see repo layout below).

**Phase 1 — Provision infra**

* IaC to create:
  + FSx for Windows File Server (specify storage capacity & throughput)
  + AWS Managed Microsoft AD (if needed) or join existing
  + DataSync agent (download OVA, deploy on-prem if needed) or use DataSync with SMB agent
  + Security Group rules and routing (ensure agent can reach AWS endpoints)
* Validate FSx reachable, AD join success, DNS resolution and SMB connectivity from a test client.

**Phase 2 — Pilot transfer(s)**

* Configure DataSync locations:
  + Source: SMB location pointing to on-prem share (credentials)
  + Destination: FSx SMB location
* Create DataSync task with options:
  + Preserve ownership & permissions (enable POSIX/SMB ACL preservation)
  + Preserve timestamps
  + Verify checksums (DataSync has verification option)
  + Exclude patterns for temp files
* Run initial sync (DataSync supports incremental syncs)
* Validate file counts, sizes, ACLs, timestamps, and application behavior.

**Phase 3 — Full migration & cutover**

* Schedule final incremental sync (quiesce writes if required).
* For minimal downtime: run final DataSync incremental, then remap clients:
  + Update DFS Namespace (if using DFS) or
  + Update DNS CNAME to new FSx endpoint, or
  + Update logon scripts / Group Policy Drive Maps
* Validate client access & permissions.
* Keep source available for fallback until TTL passes and checks pass.

**Phase 4 — Post-migration**

* Decommission or repurpose old servers.
* Enable FSx backups and lifecycle policies.
* Update runbooks, asset inventory, permissions documentation.

**Rollback plan**

* If cutover fails, revert DNS/CNAME or DFS pointers to original share.
* Keep the original source intact until rollback window closes.

**5) Automation design (Bitbucket + Jenkins)**

**Repo layout (Bitbucket)**

migrations/

├─ infra/

│ ├─ terraform/

│ │ ├─ main.tf

│ │ ├─ variables.tf

│ │ └─ modules/

├─ datasync/

│ ├─ templates/

│ │ ├─ datasync-task.json # JSON templates to create DataSync tasks via aws cli

├─ jenkins/

│ ├─ Jenkinsfile # pipeline

│ └─ scripts/

│ ├─ provision\_infra.sh

│ ├─ create\_datasync.sh

│ ├─ run\_datasync\_task.sh

│ └─ verify\_transfer.sh

└─ docs/

└─ runbooks/

**Branch strategy**

* main – production-ready IaC and pipeline
* dev – staging infra / pilot runs
* feature branches for each share or migration wave (e.g. wave-1-sales-shares)

**Jenkins pipeline (concept)**

* Pipeline triggered manually (for production runs) or by merge to main.
* Stages:
  1. Checkout
  2. Terraform Init & Plan (infra/)
  3. Terraform Apply (with manual approval for prod)
  4. Create DataSync locations & task (via AWS CLI or boto3 scripts)
  5. Start DataSync task (incremental)
  6. Poll & wait for completion, run verification script
  7. Post actions: update DNS / send notifications
  8. Mark success/fail and archive logs

**Example Jenkinsfile (Declarative) — simplified**

pipeline {

agent any

environment {

AWS\_DEFAULT\_REGION = 'us-east-1'

TF\_VAR\_env = "${params.ENV ?: 'dev'}"

}

parameters {

string(name: 'ENV', defaultValue: 'dev', description: 'env (dev/stage/prod)')

booleanParam(name: 'APPLY\_INFRA', defaultValue: false, description: 'Apply Terraform?')

string(name: 'DATASYNC\_TASK\_NAME', defaultValue: '', description: 'Name for DataSync task')

}

stages {

stage('Checkout') {

steps { checkout scm }

}

stage('Terraform Init & Plan') {

steps {

dir('infra/terraform') {

sh 'terraform init -input=false'

sh 'terraform plan -out=tfplan -input=false'

archiveArtifacts artifacts: 'tfplan', onlyIfSuccessful: true

}

}

}

stage('Terraform Apply') {

when { expression { return params.APPLY\_INFRA } }

steps {

dir('infra/terraform') {

sh 'terraform apply -input=false -auto-approve tfplan'

}

}

}

stage('Create DataSync Task') {

steps {

dir('datasync') {

// assumes create\_datasync.sh uses aws cli and returns taskArn

sh "./create\_datasync.sh ${params.DATASYNC\_TASK\_NAME} > task\_arn.txt"

archiveArtifacts artifacts: 'task\_arn.txt'

}

}

}

stage('Start Transfer') {

steps {

dir('datasync') {

sh './run\_datasync\_task.sh $(cat task\_arn.txt)'

}

}

}

stage('Verify') {

steps {

dir('jenkins/scripts') {

sh './verify\_transfer.sh $(cat ../datasync/task\_arn.txt)'

}

}

}

}

post {

success { echo 'Migration pipeline finished SUCCESS' }

failure { echo 'Migration pipeline FAILED'; mail to: 'ops@example.com', subject: "Migration failed", body: "Check Jenkins job" }

}

}

**Example scripts & commands**

* **Terraform**: create FSx (very brief snippet)

resource "aws\_fsx\_windows\_file\_system" "fsx" {

storage\_capacity = 1024 # GB

throughput\_capacity = 128 # MB/s - choose based on workload

subnet\_ids = [var.subnet\_id]

security\_group\_ids = [var.fsx\_sg]

self\_managed\_active\_directory {

domain\_name = var.domain\_name

dns\_ips = var.domain\_controllers

admin\_username = var.ad\_admin\_user

admin\_password = var.ad\_admin\_password

}

tags = { Name = "fileserver-01" }

}

* **Create DataSync location (destination)** via AWS CLI (conceptual)

# Create FSx destination location

aws datasync create-location-fsx \

--subdirectory "/share" \

--security-group-arns "arn:aws:ec2:...:sg-xxxx" \

--fsx-filesystem-arn "arn:aws:fsx:...:file-system/fs-xxxx"

* **Create SMB source location**

aws datasync create-location-smb \

--server-hostname "onprem-server.company.local" \

--subdirectory "\\share" \

--user "datasync-user" \

--mount-options Version=SMB3

* **Create DataSync task**

aws datasync create-task \

--source-location-arn <sourceArn> \

--destination-location-arn <destArn> \

--name "migrate-sales-share" \

--options '{"VerifyMode":"POINT\_IN\_TIME\_CONSISTENT","PreserveDeletedFiles":"PRESERVE","PosixPermissions":"PRESERVE", "Uid":"INT\_VALUE"}'

(Adjust options to preserve SMB/NTFS ACLs; DataSync docs detail exact flags.)

* **Start task execution**

aws datasync start-task-execution --task-arn <taskArn>

**Verification script ideas**

* Compare file counts and bytes (Get-ChildItem -Recurse | Measure-Object Length -Sum on Windows PowerShell).
* On Windows you can use robocopy in /L (list only) mode to compare counts, or use Get-FileHash samples for checksum validation of random files.
* For ACLs, run icacls on source and destination and diff results for key folders.

**6) Pilot & cutover strategy (recommended)**

* Start with a small, noncritical share (<1 TB, low activity).
* Run initial full sync (DataSync), verify, then run incremental syncs daily for 2–3 days.
* Schedule a short final delta sync in a maintenance window, update DNS/DFS pointers.
* Validate users & apps for 24–72 hours. Keep source server available and read-only for a rollback window.

**7) Common pitfalls & notes**

* **SIDs & Cross-Domain**: If FSx joins a different AD domain without trust, ACL SIDs may not map — plan for SID translation or reapply ACLs.
* **Large number of small files**: Transfers may be slow and metadata expensive. Consider parallelization, tune DataSync options, or package small files into archives if appropriate.
* **Open files**: DataSync cannot copy locked files. You may need VSS snapshots or app quiesce.
* **DNS & client caching**: Windows client DNS cache or SMB session caching may require client reboot or net use /delete operations.
* **Permissions preservation**: Verify DataSync task options for SMB/ACL preservation or use Robocopy with /SEC /COPYALL for last-mile migration if needed.

**8) Example acceptance criteria (for each share)**

* File count and total bytes within ±0.1% of source after final sync.
* Key application read/write tested successfully.
* ACLs match for a sampled list of folders and users.
* Users can access via the same path (or documented new path) with no more than X minutes of outage (specify in meeting).

**9) Templates to bring to the meeting**

* Inventory spreadsheet template (columns: server, UNC path, total size, file count, largest file, owner, ACL notes, app dependency)
* Network checklist (VPC/subnet IDs, SGs, firewall rules, proxy)
* AD checklist (domain, controllers, trust info)
* Pilot selection proposal (3 candidate shares)

**10) Quick next steps I recommend you run in the meeting**

1. Collect the inventory and define pilot shares.
2. Decide AD/JOIN strategy (AWS Managed AD vs join on-prem).
3. Approve IaC (Terraform) and Jenkins pipeline owner / credentials.
4. Schedule pilot migration window and assign verification owners.

If you want, I can:

* Produce the inventory spreadsheet template (CSV / Excel).
* Generate a full, production-ready Jenkinsfile plus the minimal Terraform module for FSx and a DataSync creation script tailored to your AWS account and AD details (I’ll need the target region, AD strategy, and one sample share from your inventory).

Which of those would you like next?

✅ Your **Bitbucket-ready repository** is ready!  
📦 [**Download fsx\_migration\_repo\_sample.zip**](sandbox://mnt/data/fsx_migration_repo_sample.zip)

This zip contains:

* **infra/terraform/** → Terraform module (mocked VPC, AD, FSx details)
* **datasync/** → Scripts to create & run DataSync tasks
* **scripts/** → PowerShell verification script
* **jenkins/** → Jenkinsfile for CI/CD pipeline
* Preconfigured with **region us-east-1**, **VPC & subnet IDs**, and **corp.example.com** AD domain

You can unzip it, push to Bitbucket, and link it to Jenkins for a fully automated FSx migration pipeline.

Would you like me to also create a **diagram of the architecture & pipeline flow** (from on-prem SMB shares → AWS DataSync → FSx → verification) so you can use it in your kickoff meeting?